**Paulo Alexandre Elias do Nascimento**

Licenciatura em Ciências da Engenharia Química e Bioquímica

# Estratégias de monitorização e automação de um SMB combinando simulação dinâmica em gProms e optimização em AMPL

Dissertação para obtenção do Grau de Mestre em
Engenharia Química e Bioquímica

Orientador: Mário Eusébio, Prof. Auxiliar, FCT/UNL

Co-orientadores: José Paulo Mota, Prof. Catedrático, FCT/UNL

Júri:

Presidente: Prof. Maria Madalena Alves Campos de Sousa Dionísio Andrade

Arguente: Dr. Ricardo Jorge Sousa da Silva

**Março, 2014**

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

**Estratégias de monitorização e automação de um SMB combinando simulação dinâmica em gProms e optimização em AMPL**

*Pela minha querida avó, Custódia Maria da Conceição Ribeiro Elias, que apesar de nos ter abandonado fisicamente, estará sempre presente na minha memória.*

# Agradecimentos

Completar este mestrado foi uma das tarefas mais difíceis mas ao mesmo tempo mais gratificante que tive até ao momento. Olhando para trás posso ver vários bons e maus momentos, muitos amigos feitos e muito trabalho para chegar até aqui. Este caminho não teria chegado ao seu término sem a ajuda e apoio de várias pessoas.

O meu primeiro agradecimento vai para o meu orientador, o professor Mário Eusébio, pois sem a sua motivação, encorajamento, ensinamentos, desafios e ajuda prestada, esta tese não teria sido concluída.

Para o Prof. José Paulo Mota, o meu agradecimento pela ajuda na disponibilização do modelo utilizado nesta tese, assim como a grande disponibilidade em esclarecer todo o tipo de questões relativas às aplicações de simulação e optimização utilizadas.

Para os meus colegas e amigos da FCT/UNL, em especial ao Pedro Camilo e à Joana Martins, por terem estado comigo nesta viagem e que tantas vezes sem eles o caminho teria sido mais difícil.

Um especial agradecimento aos meus pais e avó por toda a ajuda e todos os sacrifícios que passaram para que eu pudesse chegar a este dia, com a esperança que o dia de hoje faça com que tudo tenha valido a pena.

Por fim, o maior agradecimento possível à minha mulher e ao meu filho por toda a paciência que tiveram, por me motivarem e não me deixarem desistir deste objectivo, mesmo quando essa parecia a saída mais fácil. Vou tentar compensar o tempo perdido.

# Resumo

De modo a ultrapassar as limitações da cromatografia em Batch são utilizados processos de separação de cromatografia de leito móvel, Nesta tese é utilizado um modelo de leito móvel simulado (SMB) utilizando duas colunas.

Tendo em conta que industrialmente a composição da alimentação e as condições de operação não são constantes e tendo em consideração que as condições óptimas de operação se alteram com a temperatura, caudais e composição da mistura, é interessante conseguir simular o que acontece em diferentes cenários de operação das colunas, e encontrar o novo ponto óptimo de operação.

Uma vez que o SMB é bastante sensível às condições de operação, é importante conseguir ter acesso a este tipo de informação na interface de monitorização, automação e controlo, de forma a utiliza-la nos algoritmos de automação e controlo.

Nesta tese foi desenvolvida uma interface em Labview de modo a comunicar através de Shared Memory com o gProms (ferramenta de simulação), AMPL (ferramenta de optimização), sendo possível a visualização gráfica, na própria interface de monitorização, de resultados simulados e comparar diferentes condições de operação. Por outro lado também se recorreu aos resultados de optimização obtidos utilizando AMPL/IPOPT de forma directa na interface de modo a estabelecer as condições de automação do modelo utilizado.

Através da utilização desta interface passa a ser possível o utilizador alterar as condições de operação de uma unidade SMB de acordo com os resultados das

simulações e optimizações efectuadas, fazendo com que a produtividade seja maximizada e que o consumo de eluente face à alimentação seja minimizado.

**Palavras-chave:** SMB, Shared Memory, gProms, Labview, AMPL, monitorização, automação, controlo, simulação, optimização

---

# Abstract

In order to overcome the limitations of batch chromatography separation are used moving bed chromatography processes. In this thesis we will use a simulated moving bed model (SMB) with two columns.

Industrially the feed composition and operating conditions are not constant and the optimal operating conditions change with temperature, flow rate and composition of the mixture. Then it is interesting to simulate what happens in different column operation scenarios and find the new optimal operating point if necessary.

Since SMB is very sensitive to operating conditions, it is important to be able to access this type of information a monitoring, automation and control interface in order to use it in automation and control algorithms.

This thesis has developed an interface in Labview in order to communicate through Shared Memory with gProms (simulation tool) and AMPL (optimization tool). The graphical display, the interface itself monitoring, is possible to obtain simulated results and compare different operating conditions. Moreover it is also possible to access to optimization results obtained using AMPL / IPOPT directly in the interface, in order to establish the conditions for the automation model.

With this interface the user can change the operating conditions of a SMB unit according to the simulations results in order to maximize the productivity and minimize the consumption of eluent.

**Keywords:** SMB, Shared Memory, gProms, Labview, AMPL, monitoring, automation, control, simulation, optimization

---

# Conteúdo

# Lista de Figuras

# Lista de Tabelas

# 1 Motivação

Dos vários métodos de separação, a cromatografia é um dos mais simples e mais eficientes, permitindo separar a maioria dos compostos se a configuração da coluna, as condições de operação, a fase móvel e a fase estacionária forem as correctas.

A cromatografia em Batch é largamente utilizada em várias indústrias de larga escala devido à sua facilidade de utilização e tem como principal vantagem o facto de necessitar de pouco investimento de capital. No entanto, este tipo de operação apresenta algumas limitações, uma vez que a fase estacionária não é utilizada eficientemente e a pureza das fracções recuperadas depende bastante da selectividade do sistema utilizado.

Quando os compostos A e B a separar tem comportamento semelhante em relação ao enchimento escolhido, quando se utiliza uma coluna em Batch a zona de separação à saída é muito pequena e as quantidades que se consegue separar em cada Batch são muito pequenas e diluídas, ou seja, os picos estão (quase) sobrepostos.

Muitos dos compostos farmacêuticos são enantiómeros e um dos seus componentes ou não tem efeito terapêutico (e nesse caso pode ir com o medicamento) ou tem efeito contrário ao primeiro (ou efeitos secundários indesejáveis), pelo que é necessário conseguir fazer a sua separação.

De modo a ultrapassar as limitações da cromatografia em Batch pode-se utilizar o processo em contracorrente, com a fase móvel a circular em direcção oposta à fase estacionária. Esta utilização é o princípio operacional dos sistemas de leito móvel verdadeiro (TMB) a idealização do conceito de cromatografia de adsorção contínua em contracorrente.

Num processo em leito móvel a pureza dos compostos separados é elevada, uma vez que os perfis estão mais separados permitindo recolher fracções mais elevadas dos compostos puros.

Uma vez que o TMB é de difícil aplicação prática, o movimento contínuo da fase estacionária é simulado através da substituição do leito móvel por um conjunto de colunas de leito fixo em que os locais de entrada e recolha são periodicamente movidos na mesma direcção que o fluído segue, sendo este processo designado por leito móvel simulado (SMB).

Apesar de a pureza e o rendimento do processo SMB serem bastante superiores relativamente ao processo Batch, também implica que o processo é mais complexo e que exista a necessidade de equipamento mais versátil e de melhores ferramentas de automação, monitorização, simulação, optimização e controlo.

Uma vez que industrialmente a composição da alimentação e as condições de operação não são constantes e tendo em consideração que as condições óptimas de operação se alteram com a temperatura, caudais e composição da mistura, é interessante conseguir simular o que acontece em diferentes cenários de operação das colunas, e encontrar o novo ponto óptimo de operação.

Tendo em conta que o SMB é bastante sensível às condições de operação, é importante conseguir ter acesso a este tipo de informação na interface de monitorização, automação e controlo, de forma a utiliza-la nos algoritmos de automação e controlo.

Actualmente existe uma ferramenta de monitorização e automação do SMB, desenvolvida em Labview, uma linguagem de programação gráfica especialmente vocacionada para o desenvolvimento deste tipo de interfaces. Foram adaptados modelos anteriormente desenvolvidos em gProms, que é uma linguagem de modelação avançada de processos, para simular as condições de operação do SMB em estado transiente, e permitir a comunicação com a interface de monitorização e automação utilizando memória partilhada. Foram adaptados modelos de optimização do SMB anteriormente desenvolvidos em AMPL, que é um excelente interpretador de linguagem de alto nível que permite gerar código intermédio que é usado posteriormente por várias ferramentas de optimização, de forma a permitir a troca de informação com a interface de monitorização e controlo, utilizando chamadas ao sistema.

O objectivo principal desta tese é o desenvolvimento de ferramentas de software que permitam a comunicação entre a interface de monitorização, automação e controlo desenvolvida em Labview com as ferramentas de simulação desenvolvidas em gProms e com as ferramentas de optimização desenvolvidas em AMPL. Desta maneira passa a

ser possível a visualização gráfica, na própria interface de monitorização, de resultados simulados, comparando diferentes condições de operação. Por outro lado passará a ser possível recorrer aos resultados de optimização obtidos utilizando o AMPL e um optimizador adequado, de forma directa na interface de monitorização, automação e controlo, para deste modo estabelecer as condições de automação do SMB.

# 2 Cromatografia de leito móvel

O objectivo de um sistema com leito móvel é promover o contacto contracorrente entre as fases sólidas e líquidas, uma vez que apresenta uma alta produtividade, baixo consumo de solvente e uma grande performance de separação, o que leva a uma elevada recuperação de produto com um elevado grau de pureza, mesmo quando a selectividade é próxima da unidade ou a eficiência da fase estacionária é baixa.

No entanto, o movimento real contracorrente não é um processo eficiente devido às dificuldades causadas pelo movimento da fase sólida, pelo que se recorre ao leito móvel simulado para efectuar as separações pretendidas. [1]

## *2.1 Leito Móvel Verdadeiro (TMB)*

No TMB, as fases sólida e líquida escoam em direcções opostas (contra corrente). Sendo as afinidades dos dois compostos a separar para com a parte sólida diferentes, é possível escolher caudais para fazer um deles movimentar-se para cima e o outro para baixo.

Na figura 2.1 encontra-se representado esquematicamente um processo TMB com quatro zonas e os respectivos perfis de concentração em cada zona.

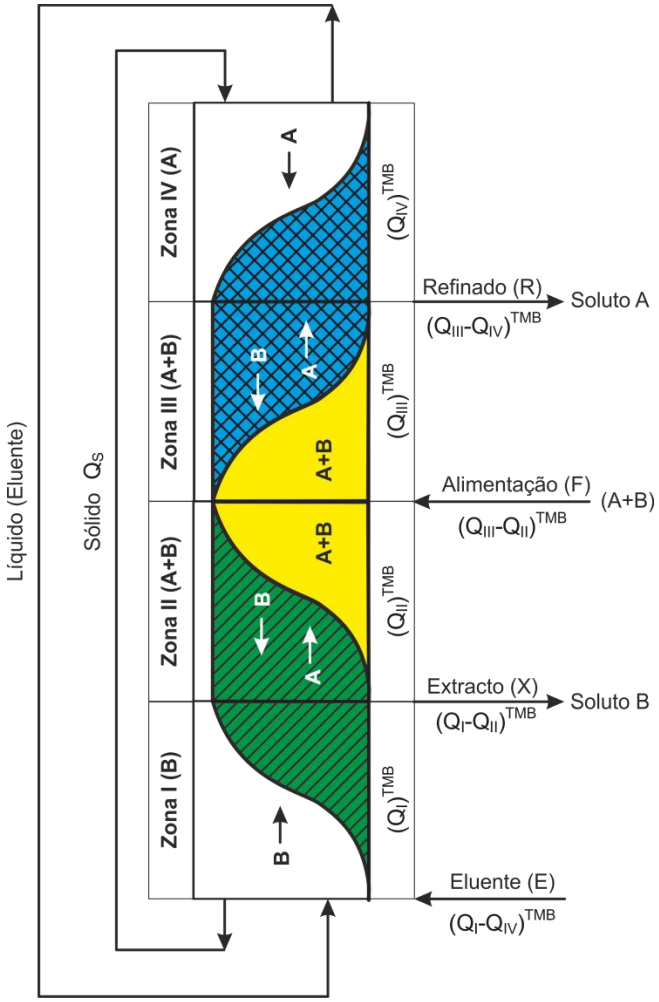


**Figura 2.1 - Representação esquemática de um processo TMB com quatro zonas e respectivos perfis de concentração.**

Este sistema requer duas entradas (uma para a alimentação e outra para o eluente) e duas saídas (uma para o extracto outra para o refinado).

O TMB clássico pode ser dividido em quatro zonas distintas, devido à adição e retirada das várias correntes. O líquido que sai na Zona IV é reciclado na Zona I e o sólido que sai na Zona I é reciclado na Zona IV.

Cada zona do TMB possui um papel específico na separação. A separação dos componentes A e B é feita nas secções II e III, onde o componente mais retido (B) tem de ser adsorvido e levado para a saída de extracto através do movimento da fase sólida, enquanto o componente menos retido (A) tem de ser eluído e levado pela fase móvel em direcção à saída de refinado.

Na secção I o sólido é regenerado por dessorção do componente mais adsorvido através da adição da corrente de eluente fresco. Por fim, na secção IV, o líquido é regenerado por adsorção do componente menos retido que não foi recolhido no refinado.

Através deste sistema é possível reciclar tanto as fases sólida e líquida para as secções IV e I respectivamente.

Com uma escolha adequada de caudais internos e da velocidade da fase sólida, a mistura de alimentação pode ser completamente separada em dois produtos puros, uma vez que só existe recolha tanto de extracto como de refinado entre as zonas I e II e III e IV, ou seja, nas pontas dos perfis de concentração.

No entanto, o movimento das partículas de adsorvente não é tecnicamente fazível devido ao atrito entre partículas e à dispersão axial que leva a que haja *backmixing*. Na prática, o movimento da fase sólida é simulado através de leitos fixos de adsorvente (colunas de cromatografia), cujos locais de entrada e saída de correntes são periodicamente movidas na mesma direcção em que segue o fluído.

## 2.2 *Leito móvel simulado (SMB)*

O SMB é um processo de separação contínuo com várias aplicações industriais e consiste numa unidade formada por um circuito de colunas de cromatografia empacotadas com um adsorvente apropriado que é dividido em quatro zonas.

No SMB as posições das linhas de alimentação, de recolha de extracto e refinado são alteradas periodicamente ao longo do leito, no sentido do fluxo da fase líquida. [2] Estas trocas são efectuadas em intervalos regulares de tempo, promovendo um movimento relativo contracorrente. A posição das linhas é alterada utilizando válvulas rotatórias que podem ser controladas via computador, de modo a que a troca das correntes seja feita de forma simultânea e eficiente.

As unidades SMB podem ter duas configurações possíveis, circuito aberto e circuito fechado, sendo que a principal diferença entre ambas é que no circuito aberto não existo recirculação do eluente.

Na unidade com circuito aberto existem duas correntes de entrada (alimentação e eluente) e três correntes de saída (extracto, refinado e eluente). Por sua vez a unidade com circuito fechado tem duas correntes de entrada (alimentação e eluente) e duas

correntes de saída (extracto e refinado). Em cada um dos casos existe rotação dos nós de entrada e saída a cada unidade de tempo $\tau$.

Na figura 2.2 pode-se verificar esquematicamente as diferenças entre ambas

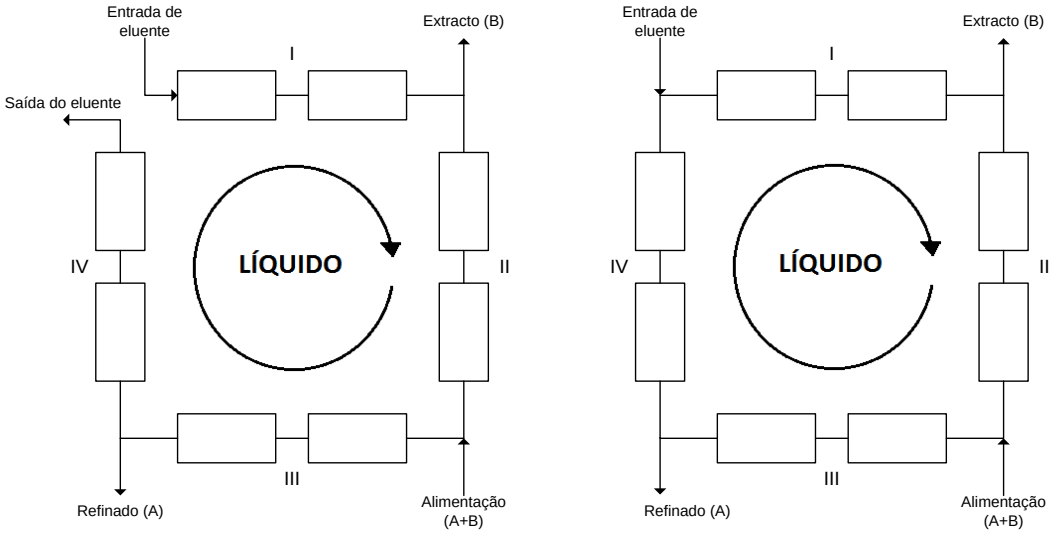


**Figura 2.2 - Configurações possíveis de uma unidade SMB com quatro zonas (duas colunas por zona) - sistema aberto à esquerda e sistema fechado à direita [2].**

Em ambas as configurações o componente com menor afinidade (A) pelo adsorvente é removido na corrente de saída do refinado enquanto o componente com maior afinidade (B) é removido na corrente de saída do extracto.

Num sistema com quatro zonas, o componente A é adsorvido na zona IV e dessorvido na zona II e a adsorção do componente B ocorre na zona III e a sua dessorção na zona I.

## 2.3 *Métodos não convencionais de SMB*

Devido à natureza periódica de um processo SMB, depois de um estado transiente inicial o sistema atinge um estado estacionário cíclico em que o processo exibe o mesmo comportamento dinâmico durante cada intervalo de comutação (*switch time* $\tau$).

Estudos recentes no campo dos processos SMB têm destacado a possibilidade de se melhorar o desempenho da separação aplicando métodos de separação não

convencionais, dos quais iremos abordar três dos mais utilizados - processo Varicol, processo PowerFeed e processo ModiCon.

### 2.3.1 Processo Varicol

Como referido anteriormente, o processo SMB implementa o processo de cromatografia em contracorrente ao avançar periodicamente tanto as correntes de entrada (eluente e alimentação) como as de saída (extracto e refinado) simultaneamente, pelo que o número de colunas em cada zona mantém-se o mesmo em cada momento, independentemente da localização das correntes de entrada e de saída.

Deste modo é possível descrever um processo SMB convencional em termos de como as colunas estão distribuídas pelas diferentes zonas como $N_I/N_{II}/N_{III}/N_{IV}$, sendo $\sum N_j = N$, onde N é um número inteiro.

No entanto, se o local das correntes sofrer uma alteração não sincronizada, teremos uma distribuição das colunas por zona que representação o comprimento médio da zona durante um ciclo completo do processo, o que faz com que $N_j$ passe a ser um número racional (apesar de N continuar a ser um número inteiro). Isto implica que o processo deixe de ser equivalente ao TMB e que agora passa a ter alguns parâmetros adicionais que são o tempo de *switch* de cada corrente de entrada e saída.

Na figura 2.3 podemos verificar um esquema simplificado de um processo Varicol, onde a posição do eluente (E), alimentação (F), extracto (X) e refinado (R) estão identificadas com setas. A altura da área sombreada é proporcional ao caudal através da coluna correspondente.

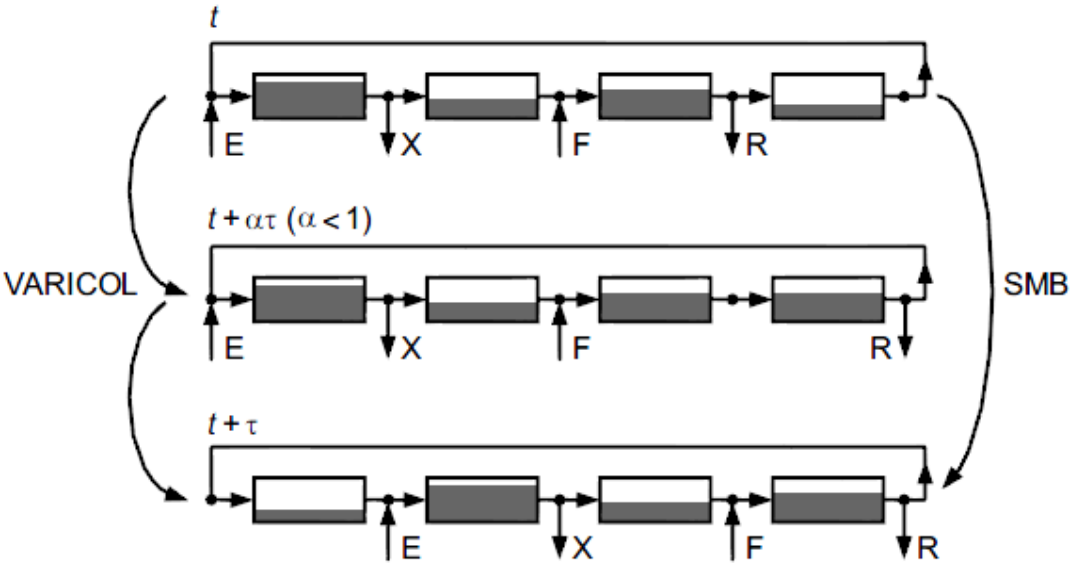


**Figura 2.3 - Esquema simplificado de um processo Varicol [4].**

O esquema do SMB convencional mantém a configuração constante durante todo o intervalo de *switch* (τ), em que a localização de todas as correntes avança uma coluna na direcção do fluído.

No processo Varicol as correntes são mudadas de forma assíncrona. Por exemplo, após uma fracção $\alpha < 1$ do intervalo de *switch*, a posição da corrente de refinado é movida para a frente uma coluna, dando origem à configuração no meio da figura, aumentando uma coluna à zona III, removendo uma coluna à zona IV, mas mantendo os caudais das zonas I, II e III inalteráveis. No fim do intervalo de *switch* todas as correntes moveram-se uma coluna como no esquema do SMB convencional.

Neste esquema, o SMB é um processo com 4 colunas onde a configuração das colunas é 1/1/1/1 no processo convencional. No processo Varicol seria descrito como $1/1/(2-\alpha)/\alpha$ e ainda manteria $\sum N_j = 4$, como no processo convencional.

Este processo foi patenteado no ano 2000 [5] e é comercializado desde então pela empresa Novasep, sendo a sua principal aplicação a separação de enantiómeros.

A principal vantagem deste processo é possibilidade de permitir uma mais eficiente e flexível alocação da fase estacionária a cada uma das quatro zonas de acordo com as necessidades da tarefa de separação, fazendo com que a separação seja mais eficiente e permitindo que a operação seja feita com um menor número de configurações de colunas e, ainda assim, manter ou até aumentar a produtividade do processo relativamente às unidades SMB convencionais. [6-9]

### 2.3.2 Processo PowerFeed

Outros parâmetros que podem ser manipulados durante o intervalo de *switch* de um processo SMB são os caudais de cada zona.

Todos os estudos deste processo são teóricos ou numéricos com a excepção de esquema simples de PowerFeed para separação de enantiómeros baseado numa variação da alimentação em 3 passos. [10] Este processo mostrou que ou a produtividade aumentava ou o consumo de solvente era menor em comparação com o processo SMB convencional.

Na figura 2.4 podemos verificar um esquema simplificado de um processo PowerFeed, onde a posição do eluente (E), alimentação (F), extracto (X) e refinado (R) estão identificadas com setas. A altura da área sombreada é proporcional ao caudal através da coluna correspondente.

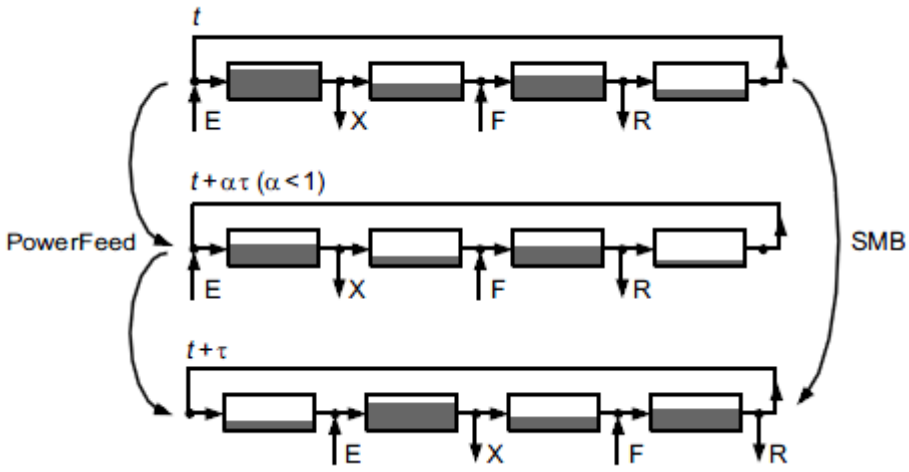


**Figura 2.4 - Esquema simplificado de um processo PowerFeed. [4]**

O esquema do SMB convencional mantém a configuração constante durante todo o intervalo de *switch* (τ), em que a localização de todas as correntes avança uma coluna na direcção do fluído.

No processo PowerFeed, descrito pelo passo intermédio, os caudais variam numa fracção do intervalo de switch mas a localização das correntes de entrada e de saída só se movem no fim do intervalo, como no caso do SMB convencional.

Este processo foi patenteado pela primeira vez em 1992 por Kearney e Hieb. [11]

### 2.3.3 Processo ModiCon

Como no processo SMB convencional o processo ModiCon é sincronizado e é caracterizado por ter um caudal constante em cada zona. O aumento de performance é devido a uma alteração da concentração da alimentação após o intervalo de *switch*.

Como se pode verificar na figura 2.5, o processo ModiCon consiste em dividir os intervalos de *switch* em vários passos constantes com diferentes concentrações da alimentação.

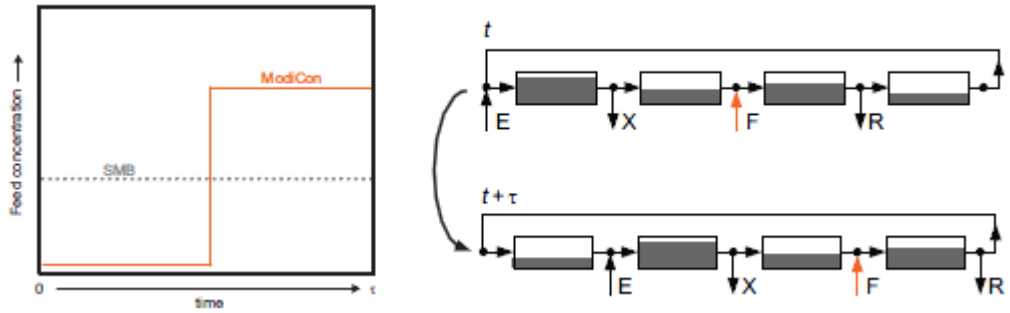


**Figura 2.5 - Esquema simplificado de um processo ModiCon. [4]**

No lado direito da figura 2.5 podemos ver que o comportamento da troca do local as correntes é exactamente igual ao do SMB convencional, ou seja, todas as correntes são simultaneamente movidas uma coluna para a frente na direcção do fluído.

O gráfico no lado esquerdo mostra a evolução da concentração da alimentação após o switch. No caso exemplificado na figura, numa parte do intervalo de switch a alimentação é nula ($c_F$=0) e, após esse período, é adicionado alimentação em que a concentração é o dobro da concentração do processo SMB convencional equivalente, o que na prática faz com a tenha sido adicionada a mesma quantidade de alimentação ao processo em ambos os casos.

Este processo aumenta tanto a produtividade, a concentração do produto e reduz a quantidade de solvente consumido. No entanto, uma das desvantagens deste processo é o facto de apenas funcionar para separações não lineares e apenas é eficiente para separações em que a concentração da alimentação é limitada por questões técnicas e não pela solubilidade dos compostos no solvente utilizado. [12]

O processo ModiCon foi patenteado em 2004 [13] e é comercializado pela empresa Knauer.

## 2.4 *Modelação de um SMB*

A modelação e simulação de processo é extremamente importante em maior parte dos processos em engenharia química e o processo SMB não é excepção, de modo a encontrar um ponto de operação óptimo de entre as quase infinitas escolhas possíveis.

Um modelo SMB é constituído pelo modelo da coluna de cromatografia (descrita no capítulo 2.2 e pelas equações dos nós.

Os modelos das colunas descrevem o balanço de massas para cada coluna de cromatografia enquanto as equações dos nós ligam os vários modelos individuais de modo a formar a totalidade do sistema de leito móvel.

Para efeitos de modelação e simulação o comportamento inicial do SMB tem pouca relevância, sendo o regime de estado estacionário o que é utilizado uma vez que é uma abordagem mais rápida e simples. O estado estacionário é geralmente atingido após um reduzido número de ciclos.

Existem vários modelos para descrever e/ou prever o desempenho de um processo SMB.

Existem geralmente duas abordagens para a modelação de um processo SMB: simular directamente num SMB verdadeiro, tendo em consideração o comportamento periódico do sistema; ou assumir a equivalência entre os processos SMB e TMB.

### 2.4.1 Modelo do SMB verdadeiro

A abordagem do SMB verdadeiro analisa cada subsecção individualmente e, analisando o esquema da figura 2.2 podemos começar por definir o balanço geral:

$$Q_E + Q_F = Q_X + Q_R \tag{2.1}$$

Tendo em consideração os nós do processo, a figura 2.6 apresenta a representação esquemática de um nó genérico ligando duas colunas consecutivas (*j-1* e *j*) de uma unidade SMB.

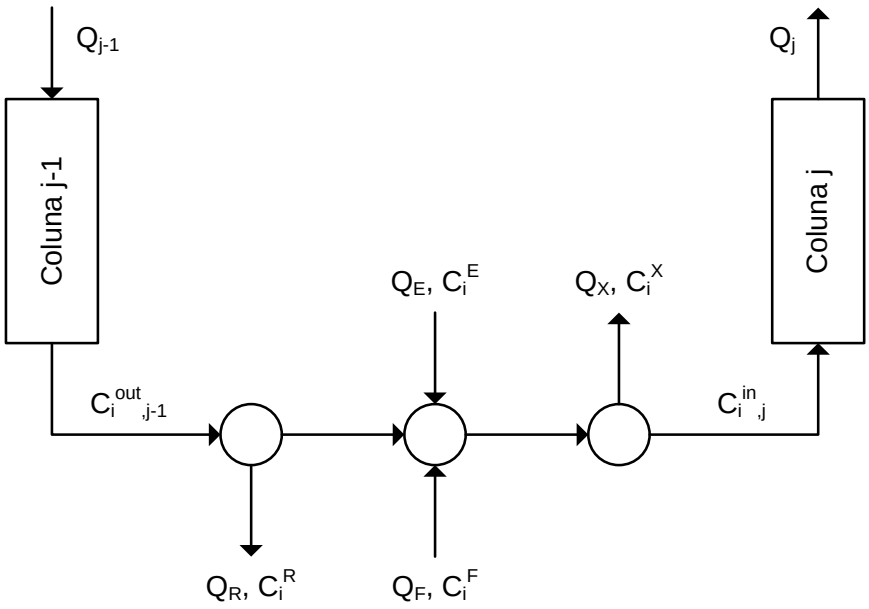


**Figura 2.6 - Representação esquemática do nó que liga as colunas j-1 e j de uma unidade SMB. [4]**

Onde $c_i$ é a concentração do soluto na fase líquida, $Q$ o caudal volumétrico e E, $X$, $F$ e $R$ são respectivamente o eluente, extracto, alimentação e refinado.

A sequência apresentada na figura 2.6 em que as linhas de entrada e saída estão ligadas ao nó de modo a que permita que certas zonas deixem temporariamente de existir, de modo a evitar mistura de correntes.

As equações abaixo têm uma aplicação geral em situações em que as linhas não se cruzam entre si e que as zonas adjacentes não deixam de existir simultaneamente.

Assim, o balanço do soluto no nó pode ser escrito do seguinte modo:

$$\begin{cases} Q_{III}c_{i,j}^{in} = Q_{II}c_{i,j-1}^{out} + (Q_{III} - Q_{II})c_i^F & (nó\ da\ alimentação) \\ Q_I c_{i,j}^{in} = Q_{IV}c_{i,j-1}^{out} + (Q_I - Q_{IV})c_i^E & (nó\ do\ eluente) \\ c_{i,j}^{in} = c_{i,j-1}^{out} & (restantes\ casos) \end{cases}, \quad (2.2)$$

Onde $Q_I$, $Q_{II}$, $Q_{III}$, *e* $Q_{IV}$ são os caudais nas quatro zonas do SMB e são determinados pelos balanços globais aos nós:

$$Q_I = Q_{IV} + Q_E \tag{2.3}$$

$$Q_{II} = Q_I - Q_X \tag{2.4}$$

$$Q_{III} = Q_{II} + Q_F \tag{2.5}$$

$$Q_{IV} = Q_{III} - Q_R \tag{2.6}$$

As linhas de saída são definidas por:

$$\begin{cases} Q_X c_i^X = Q_I c_{i,j-1}^{out} & (nó\ do\ extracto) \\ Q_R c_i^R = Q_{III} c_{i,j-1}^{out} & (nó\ do\ refinado) \end{cases} \tag{2.7}$$

A operação cíclica do processo SMB é conseguida movendo as linhas de entrada e de saída uma coluna (na direcção da fase líquida) todas as unidades de tempo $\tau$, onde $\tau$ é o intervalo de troca.

Matematicamente é expresso do seguinte modo [4]:

$$\Omega_{(t)} = \Omega_{(t_j)}\,, t_j = [t - (j-1)\tau]\ mod\ N_\tau \tag{2.8}$$

Onde i é o conjunto de variáveis de entrada para a coluna *j*, que inclui $Q_i$ e o estado (aberto ou fechado) das quatro linhas de entrada/saída ligadas ao nó de entrada, "*mod*" define o operador habitual: *a mod b ≡ a – b int (a/b).*

De notar que a equação 2.8 impõe uma periodicidade $N\tau$ em *j(t)* em que:

$$j - 1(t) = j(t + \tau) \tag{2.9}$$

### 2.4.2 Equivalência entre o TMB e o SMB

Onde o TMB, após um estado transiente inicial, atinge um estado estacionário real, o SMB atinge um estado cíclico periódico onde o processo exibe a mesmo comportamento dinâmico após cada intervalo de troca. Apesar das diferenças, as duas unidades podem ser consideradas equivalentes, com o grau de igualdade a aumentar com o número de colunas que a unidade SMB apresentar. Em estudos já efectuados [14]

verificou-se que para processo com duas colunas por zona a equivalência é bastante aproximada.

A analogia entre as duas unidades é feita através da manutenção de uma constante da relação entre a velocidade do líquido em relação à velocidade do sólido.

$$v_j^{TMB} = v_j^{SMB} - v_s \tag{2.10}$$

Em que *vj* e *vs* são respectivamente as velocidades da fase líquida e sólida.

A velocidade do sólido nos modelos TMB está também relacionada com o intervalo de *switch* $\tau$ do modelo SMB, onde *L* é o comprimento de uma coluna da unidade de SMB:

$$v_s = \frac{L}{\tau} \tag{3.11}$$

Uma vez que o caudal volumétrico pode ser descrito como:

$$Q_s = (1-\varepsilon)Av_s \tag{2.12}$$

$$Q_j^{SMB} = \varepsilon A v_j^{SMB} \tag{2.13}$$

Então a equivalência entre as fases sólidas e líquida podem ser reescritas como:

$$\tau = \frac{(1-\varepsilon)V}{Q_s} \tag{2.14}$$

$$Q_j^{SMB} = Q_j^{TMB} + \frac{\varepsilon}{1-\varepsilon} Q_s \tag{2.15}$$

Com a equivalência com o comprimento de cada zona:

$$N_j V = V_j^{TMB} \tag{2.16}$$

Em que $V = AL$ é o volume da coluna SMB, $V_j^{TMB}$ é o volume da zona $j$ no TMB e $N_j$ é o número de colunas na zona $j$ do SMB.

**Teoria do Triângulo**

A analogia mais comum entre os processos TMB e SMB é um método chamado de Teoria do Triângulo. Este metido utiliza um modelo TMB simplificado baseado na teoria do equilíbrio que ignora a dispersão axial e a resistência aos efeitos de transferência de massa, ou seja, assume que a coluna tem uma eficiência infinita.

Com estas assumpções, se considerarmos uma separação binária, a equação do balanço de massas para o soluto $i$ na zona $j$ da unidade TMB é dada por:

$$\varepsilon \frac{\partial c_{i,j}}{\partial t} + \beta \frac{\partial q_{i,j}}{\partial t} + \frac{Q_j}{\varepsilon A} \frac{\partial c_{i,j}}{\partial z} = 0 \qquad (0 < z < L) \tag{2.17}$$

Onde $A$ é a área de secção recta da coluna, $L$ é um comprimento da coluna, $z$ a coordenada axial ao longo da coluna. O símbolo $t$ representa a coordenada do tempo, ε a porosidade entre partículas na coluna, $Q_j$.o caudal volumétrico do líquido, $c$ e $q$ as concentrações das fases líquidas e sólida respectivamente.

Utilizando parâmetros adicionais, $m_j$, denominado rácio de caudais, em que define o ratio entre o caudal da fase líquida sobre o caudal da fase sólida em cada zona, podemos utilizar esta analogia em termos de parâmetros operacionais numa unidade SMB equivalente:

$$m_j = \frac{Q_j^{TMB}}{Q_s} = \frac{Q_j^{TMB}\tau}{(1-\varepsilon)V} - \frac{\varepsilon}{1-\varepsilon} \tag{2.18}$$

Para recuperar o soluto mais adsorvido (B) na linha do extracto e o menos retido (A) na linha do refinado, os parâmetros operacionais deverão ter os valores apropriados de modo a garantir determinadas condições:

- Na zona I, a espécie mais retida (B) deve seguir o movimento da fase líquida de modo a que a fase sólida seja regenerada;
- Nas Zonas II e III a espécie mais retida (B) deve migrar na direcção oposta à da fase líquida e a espécie menos retida (A) deve seguir na direcção da fase líquida;
- Na zona IV, A deve seguir na direcção oposta à da fase líquida de modo a que o eluente seja eficientemente regenerado.

Matematicamente estas três condições podem ser expressas por:

$$m_1 > \left(\frac{\partial q_B^*}{\partial c_B}\right)_I, \left(\frac{\partial q_A^*}{\partial c_A}\right)_{II} < m_2 < \left(\frac{\partial q_B^*}{\partial c_B}\right)_{II}, \left(\frac{\partial q_A^*}{\partial c_A}\right)_{III} < m_3 < \left(\frac{\partial q_B^*}{\partial c_B}\right)_{III}, m_4 > \left(\frac{\partial q_A^*}{\partial c_A}\right)_{IV} \quad (2.19)$$

Onde $(\partial q_i/\partial c_i)$ é o declive médio da isotérmica de adsorção da espécie *i* na zona *j*. Considerando concentrações de alimentação baixas, ou seja, uma isotérmica de adsorção linear ($q_i=K_ic_i$), as condições necessárias e suficientes para a completa separação dos dois componentes A e B, em que $K_B > K_A$, consiste nas seguintes desigualdades [15]:

$$K_B < m_1 < \infty, K_A < m_2 < m_3 < K_B, \frac{-\varepsilon_p}{1-\varepsilon_p} < m_4 < K_A \quad (2.20)$$

Estas condições vão definir uma região onde vai existir a completa separação de A e de B no espectro do modelo baseado na equivalência entre SMB e TMB, que assume o equilíbrio ideal.

Habitualmente só a região triangular composta pelo plano ($m_2$,$m_3$) é usada para representação gráfica (figura 2.7) mas é preciso ter em conta que apenas se aplica tendo em conta que as condições no plano ($m_1$,$m_4$) também são cumpridas e que o ponto de operação é escolhido na região onde existe completa regeneração (figura 2.8).

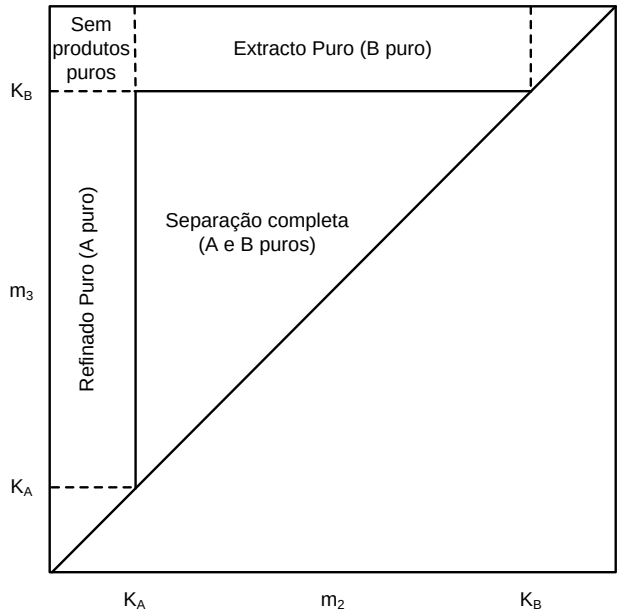


**Figura 2.7 - Região de separação completa no espaço operacional m2,m3.**

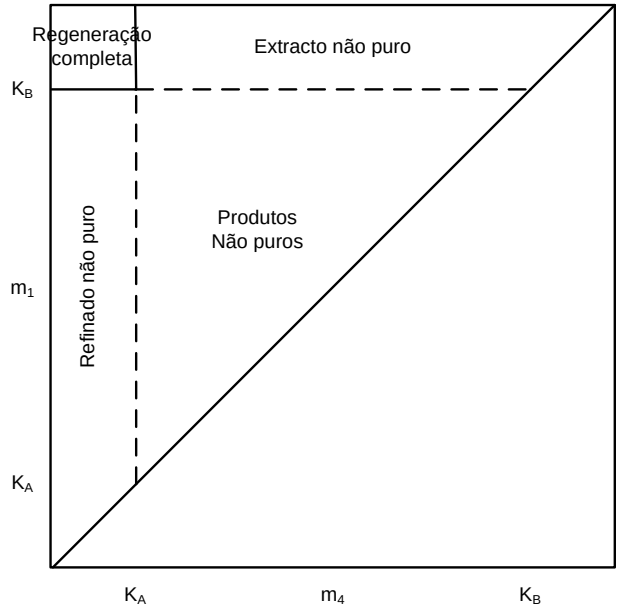


**Figura 2.8 - Região de regeneração completa no espaço operacional m1,m4.**

Por outro lado, no caso de isotérmicas de adsorção não lineares, a região da completa separação depende da concentração dos compostos na alimentação e o triângulo "encolhe" com o aumento da concentro da alimentação, $c_F$, como se pode verificar na figura 2.9.

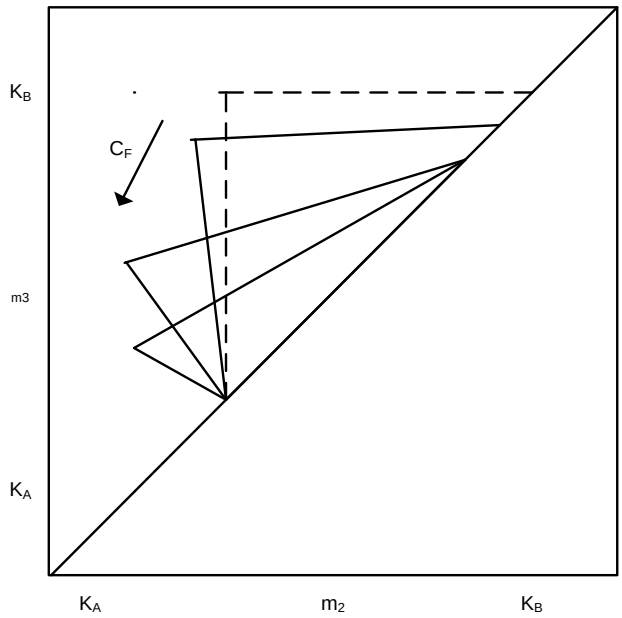


**Figura 2.9 – Comportamento da região de completa separação no caso de uma isotérmica de adsorção não linear obtida através da Teoria do Triângulo com o aumento da concentração da alimentação – CF.**

Na prática, a performance de um processo SMB é habitualmente avaliada utilizando a produtividade (maximizada) e o consumo de eluente (minimizado).

$$Produtividade = \frac{Q_F c_F}{NV} = \frac{(m_3 - m_2)(1-\varepsilon)c_F}{N\tau} \quad (2.21)$$

$$Consumo\ de\ eluente = 1 + \frac{Q_E}{Q_F} = 1 + \frac{m_1 - m_4}{m_3 - m_2} \quad (2.22)$$

Através das equações 2.21 e 2.22 verifica-se que através da constante do intervalo de troca, $\tau$, concentração da limentação, $c_F$, e a diferença entre $m_1$ e $m_4$, a produtividade aumenta e o consumo de eluente diminui com a distância à diagonal no plano $m_2$-$m_3$.

Assim, o ponto de operação óptimo que maximiza a produtividade e diminui o consumo de eluente é o vértice da região triangular da separação completa. Na prática é utilizado um ponto de operação próximo desse ponto óptimo ma dentro da região da separação completa, de modo a garantir uma margem de segurança para a operação.

A teoria do triângulo é utilizada com dois objectivos diferentes - para explicar resultados experimentais ou determinar condições de operação óptimas.

## 2.5 *Sistema de duas colunas para separação de dois compostos*

Apesar da implementação clássica de um processo SMB ser composto por quatro zonas com um número inteiro de colunas por zona, têm sido estudadas algumas alternativas com um menor número de zonas, desde os SMB com três zonas estudado por Chin e Wang [16] ou os processos SMB com duas zonas propostos por Lee [17], ou Jin e Wankat [18] cujos resultados mostraram ser possível melhorar a pureza dos produtos e quantidades recolhidas, apesar de ser necessário uma maior quantidade de eluente utilizada em relação ao processo SMB com quatro zonas.

Recentemente foi desenvolvido um processo de cromatografia semi-contínuo com um nó flexível de modo a aproveitar tanto os benefícios tanto do processo Batch como do processo SMB [19]. Uma das vantagens da utilização do processo abaixo descrito é o facto de ser facilmente realizado, pois a configuração mais simples não tem passos de recirculação e precisa apenas de duas bombas, uma para adicionar a alimentação e outra para adicionar o eluente ao sistema, enquanto os caudais de líquido retirado do sistema são controlados pelo balanço de massa utilizando duas válvulas.

No trabalho efectuado por Ricardo Silva [20] foi efectuada a comparação deste modelo em comparação com a cromatografia em Batch com reciclo num caso com adsorção não linear e onde se verificou que se consumia menos eluente para uma mesma produtividade, cuja representação esquemática encontra-se na figura 2.10.

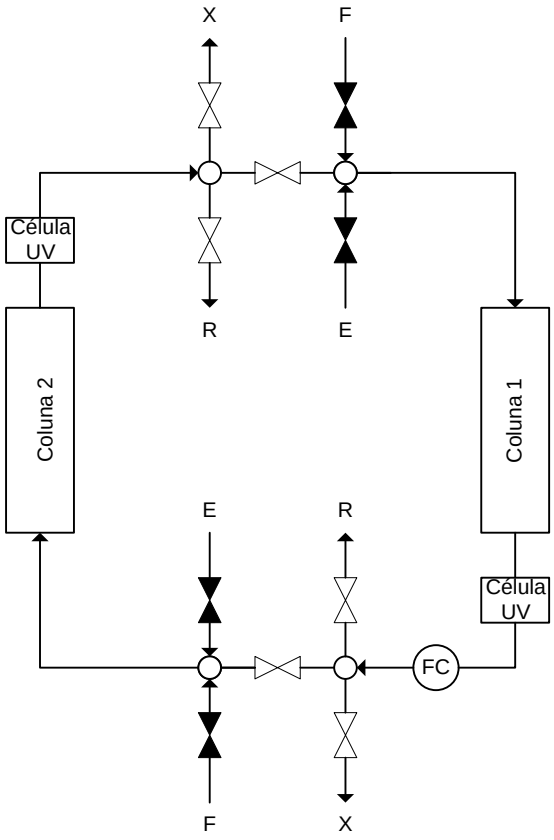


**Figura 2.10 - Representação esquemática de um sistema de duas colunas para separação de dois compostos.**

O modelo da coluna utilizado no trabalho citado e também nesta tese é uma boa aproximação da dinâmica de uma coluna de cromatografia se a isotérmicas de adsorção forem lineares (Lei de Henry) e se a transferência de massa for controlada pelas difusão molecular.

O balanço de massas para a coluna de cromatografia é dado por:

$$\frac{\partial c_i}{\partial \theta} + \beta \frac{\partial q_i}{\partial \theta} = \frac{\tau Q}{\varepsilon V_C}\left(\frac{1}{Pe_i}\frac{\partial^2 c_i}{\partial x^2} - \frac{\partial c_i}{\partial x}\right) \ , \ para\ 0 < x < 1 \qquad (2.23)$$

Onde $i$ é o índice do soluto, $\theta = t/\tau$ e $x = z / L$, sendo $\theta$ e $x$ respectivamente as coordenadas temporais e axiais adimensionais, $c_i$ e $q_i$ respectivamente as concentrações da fase líquida e sólida, $\beta = (1-\varepsilon)/\varepsilon$ o rácio da fase, $V_C$ o volume geométrico da coluna, $Q$ o caudal da fase líquida e $Pe$ o número de Péclet aparente.

A equação 2.23 está sujeita às condições-fronteira habituais:

$$c_i - \frac{1}{Pe_i}\frac{\partial c_i}{\partial x} = c_i^{in} \quad para\ x = 0 \tag{2.24}$$

$$\frac{\partial c_i}{\partial x} = 0 \quad para\ x = 1 \tag{2.25}$$

As concentrações do soluto $i$ nas fases líquidas e sólida, $c_i$ e $q_i$ respectivamente, estão relacionadas com as isotérmicas de adsorção.

$$q_i = f(c_1, c_2, \dots, c_n)\ \ i = 1, 2, \dots, n_{solutos} \tag{2.26}$$

Como as isotérmicas de adsorção são lineares, as derivadas no tempo $\partial q_i/\partial\theta$ são dadas através das constantes de Henry correspondentes.

Este modelo pode também ser utilizado como uma razoável aproximação para isotérmicas de adsorção não lineares, em que $\partial q_i/\partial\theta$ é calculado através de:

$$\frac{\partial q_i}{\partial\theta} = \sum_{k=1}\left(\frac{\partial q_i^*}{\partial c_k}\right)\left(\frac{\partial c_k}{\partial\theta}\right) \tag{2.27}$$

# 3 Comunicação entre processos

Os vários sistemas operativos fornecem mecanismos para facilitar a comunicação e a partilha de dados entre aplicações. As actividades permitidas por estes mecanismos são chamadas de *Interprocess Communications* (IPC).

Numa comunicação entre processos, cada um dos processos pode funcionar tanto como cliente ou servidor. Um cliente é uma aplicação ou processo que solicita um serviço de outras aplicações ou processos. Um servidor é uma aplicação ou processo que responde ao pedido do cliente. Algumas aplicações podem agir tanto como cliente como servidor, dependendo da situação.

## *3.1 Mecanismos de comunicação*

Existem vários mecanismos de comunicação entre processos disponíveis que poderão ser usados individualmente ou simultaneamente, dependendo da necessidade das aplicações envolvidas.

Para se decidirem que mecanismos deverão ser utilizados devem ser tomados em conta alguns aspectos, tal como se a aplicação deve comunicar com outras aplicações a correr noutros computadores ou é suficiente comunicar com aplicações no mesmo computador, se a aplicação deve comunicar com outras aplicações a correr noutros computadores com outro sistema operativo, ou se a aplicação deve comunicar directamente com as outras aplicações ou aguardar por indicação do utilizador.

O Windows suporta vários mecanismos de comunicação entre processo, tendo as suas vantagens e desvantagens.

Alguns dos mecanismos fornecidos e/ou suportados pelo Windows, que serão abordados de forma mais detalhada, são [22]:

- Clipboard (Área de transferência)
- File Mapping (Shared Memory);
- Mailslots;
- Named Pipes;
- TCP/IP

**Clipboard**

O clipboard é um recurso utilizado por um sistema operativo para o armazenamento de pequenas quantidades de dados para transferência de dados entre documentos ou aplicações.

Quando o utilizador efectua uma operação de "cortar"/"copiar" ou "colar" numa aplicação, a aplicação põe os dados seleccionados no clipboard num ou mais formatos padrão ou definidos pela própria aplicação. Qualquer outra aplicação poderá aceder aos dados depositados no clipboard, escolhendo de entre os formatos disponíveis. Este tipo de comunicação é disponibilizado pelo Windows e tem os seguintes atalhos:

- CTRL+C para copiar dados para o clipboard;
- CTRL+X para cortar dados para o clipboard;
- CTRL+V para colar dados a partir do clipboard.

A diferença entre o CTRL+C e o CTRL+X é que o primeiro mantém os dados na aplicação de origem enquanto o segundo apaga os mesmos após estes terem sido colados na aplicação de destino.

O clipboard pode ser utilizado para comunicação entre aplicações no mesmo computador ou em computadores distintos.

As funções principais da comunicação através do clipboard são o "OpenClipboard", "EmptyClipboard", "SetClipboardData" e "CloseClipboard".

- A função "OpenClipboard" abre o clipboard para iniciar a comunicação.
- A função "EmptyClipboard" esvazia a informação presente no clipboard.
- A função "SetClipboardData" é chamada quando se pretende escrever informação no clipboard.
- Por fim a função "CloseClipboard" é utilizada para terminar a comunicação e fechar o clipboard.

**Mailslots**

Os *mailslots* fornecem comunicação unidireccional. A mensagem enviada pelo cliente é sempre adicionada ao *mailslot*. O *mailslot* guarda a mensagem até que o servidor a tenha lido. O processo pode ser tanto um servidor como um cliente, pelo que é possível comunicação em ambos os sentidos caso se usem múltiplos *mailslots*.

Um cliente *mailslot* pode enviar a mensagem para uma *mailslot* no mesmo computador, para uma *mailslot* noutro computador ou para todas as *mailslots* com o mesmo nome em todos os computadores numa rede.

O *mailslot* é geralmente um mecanismo que é utilizado por permitir às aplicações enviarem e receberem mensagens curtas e, principalmente, por permitir enviar mensagens simultaneamente para vários computadores ligados em rede.

A comunicação por *mailslots* utiliza três funções principais, sendo estas "CreateMailslot", "GetMailslotInfo" e "SetMailslotInfo".

- A função "CreateMailslot" inicia a comunicação por *mailslot*.
- A função "GetMailslotInfo" é chamada para se receber informação do *mailslot*.
- A função "SetMailslotInfo" é chamada quando se pretende enviar informação através de *mailslot*.

**Pipes**

Existem dois tipos de *pipes* para comunicação nos dois sentidos – *anonymous pipes* e *named pipes*.

Os *anonymous pipes* permitem que processos relacionados troquem informação entre si. Para ser trocada informação nos dois sentidos é necessário criar dois *pipes*: O processo 1 escreve dados num *pipe* utilizando a sua função de escrita enquanto o processo 2 lê os dados escritos no *pipe* através da sua função de leitura. No outro *pipe* é feito o mesmo procedimento, sendo que o processo 2 escreve e o processo 1 lê os dados. Os *anonymous pipes* só podem ser utilizados entre processos relacionados entre si e que estejam no mesmo computador.

Por seu lado os *named pipes* são utilizados para transferir dados entre processos que não estão relacionados entre si e processos que estejam em computadores diferentes. Geralmente o servidor do *named pipe* cria um *pipe* com um nome que é comunicado aos clientes. O cliente do *named pipe* pode ter acesso aos dados, sujeito às restrições de acesso especificados pelo servidor. Assim que ambos se tenham conectado ao *pipe*, estes podem trocar dados utilizando funções de escrita e leitura.

A comunicação por *named pipes* baseia-se em cinco funções principais, sendo estas "CreateFile", "CallNamedPipe", "ConnectNamedPipe", "CreateNamedPipe" e "DisconnectNamedPipe".

- A função "CreateFile" cria um *named pipe*.
- A função "CallNamedPipe" conecta-se a um *named pipe*, escreve e lê dele e depois fecha-o.
- A função "ConnectNamedPipe" coloca o processo que é o servidor de um *named pipe* a aguardar que um processo cliente se conecte a uma instância desse *pipe*.
- A função "CreateNamedPipe" cria uma instância de um *named pipe* e retorna um *handle* para as operações realizadas nesse *pipe*. O processo cliente conecta-se a esse *pipe* utilizando as funções "CreateFile" ou "CallNamedPipe".
- A função "DisconnectNamedPipe" termina a ligação do processo cliente com o servidor do *named pipe*.

**File mapping**

A comunicação por *file mapping* permite a um processo tratar o conteúdo de um ficheiro como se estivesse num bloco de memória. O processo pode utilizar apontadores para examinar e modificar o conteúdo do ficheiro.

O *file mapping* é o processo de comunicação mais rápido de todos os referidos mas apenas pode ser utilizado entre processos dentro do mesmo computador.

O primeiro processo cria o objecto de *file mapping* chamando a função "CreateFileMapping" que cria um *handle* e um nome para o objecto. Posteriormente é chamada a função "MapViewOfFile", que mapeia o objecto e devolve um apontador para o local onde se encontra o objecto.

Após o ficheiro estar mapeado é utilizada a função "CopyMemory" de modo a escrever uma string que possa ser acedida por outros processos.

Quando o processo já não necessitar de aceder ao objecto *file mapping*, deve ser chamada a função "CloseHandle" para libertar a secção de ficheiro que este objecto utiliza.

Um segundo processo pode aceder à *string* escrita na memória pelo primeiro processo chamando a função "OpenFileMapping" especificando o mesmo nome para o objecto mapeado definido pelo primeiro processo.

Posteriormente pode ser utilizada a função "MapViewOfFile" de modo a obter um apontador para o local onde se encontra o objecto, obtendo a informação escrita pelo primeiro processo.

Este procedimento pode ser repetido para mais processos, sendo que quando dois ou mais processos acedem ao mesmo objecto de *file mapping*, cada um deles recebe um apontador para a memória que pode utilizar para ler ou modificar o conteúdo do ficheiro. Os processos devem então utilizar um objecto de sincronização de modo a que seja evitada corrupção de dados num ambiente *multitasking*.

**TCP/IP**

O "Transmission Control Protocol" (TCP) que é um dos protocolos principais da "Internet Protocol Suite" (IP), pelo que normalmente ao conjunto desses protocolos é chamado TCP/IP.

O IP funciona através da troca de partes de informação chamadas *packets*, que são sequências de octetos (conjuntos de 8bits) e consiste num *header* seguido de um *body*, em que o *header* descreve a origem, fonte e controlo de informação do *packet* e o *body* contem a conteúdo da informação a transmitir.

O TCP funciona como um meio de comunicação entre uma aplicação e o protocolo de internet (IP). Quando é necessário o envio de dados através da internet utilizando um IP, a aplicação faz um pedido ao TCP e este trata da conversão da informação para octetos de forma fiável, ordenada e livre de erros.

Assim que o TCP tiver voltado a juntar todos *packets*, a informação é enviada para a aplicação de destino.

O TCP permite a comunicação entre computadores ligados em redes locais, intranet ou através da internet. Actualmente é utilizado pela maior parte das aplicações de internet mais conhecidas, tal como World Wide Web (WWW), e-mail e aplicações de transferência de ficheiros peer-to-peer.

Ao contrário de outros protocolos que utilizam IP, o TCP garante que a informação que é enviada é recebida exactamente do mesmo modo.

O TPC pode ser dividido em três fases, a fase de estabelecimento das ligações, a fase de transferência de dados e a fase de terminação das ligações.

Na figura 3.1 pode-se verificar um diagrama de estado de uma comunicação via TCP/IP [23] em que são indicadas as várias funções utilizadas para a mesma.

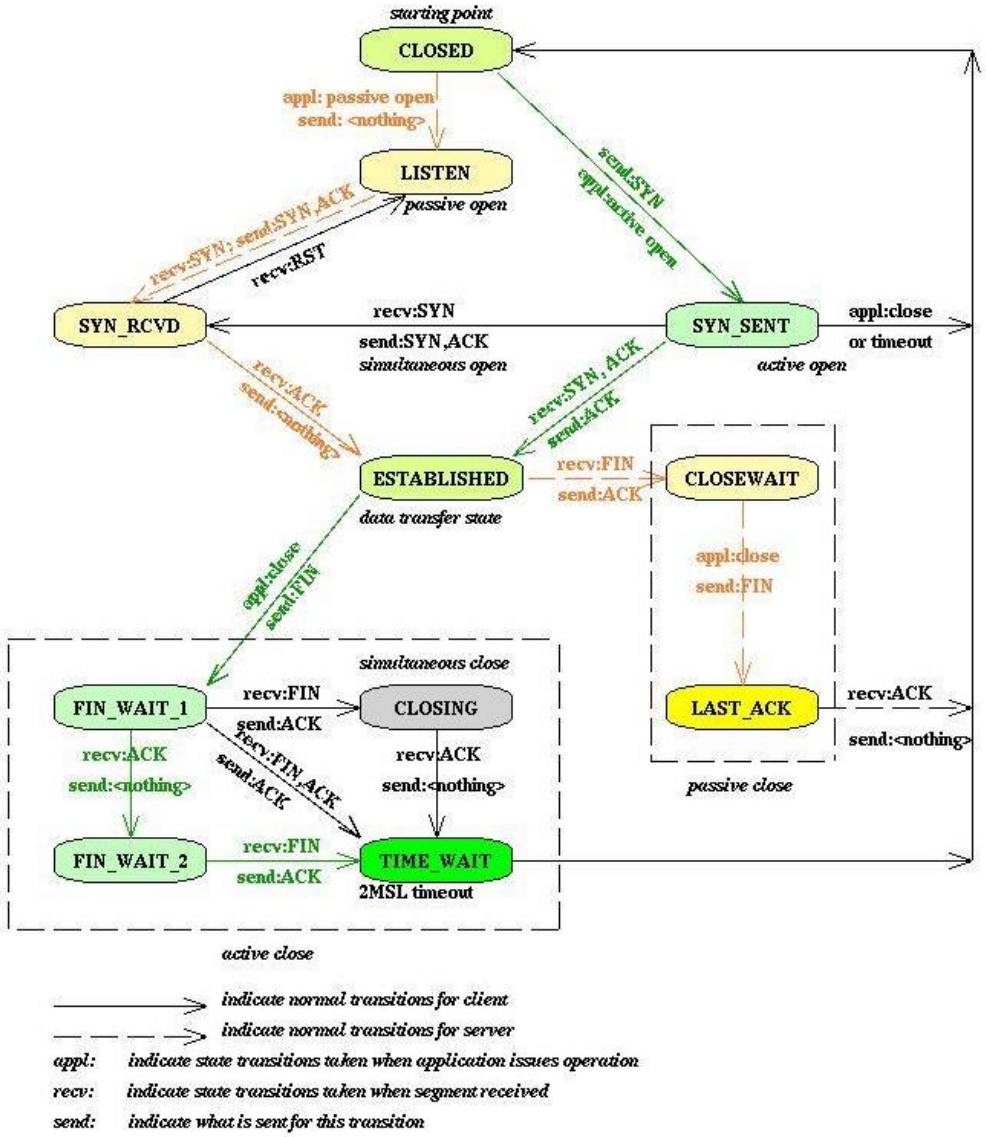


**Figura 3.1- Diagrama de estado de uma comunicação via TCP/IP.**

## 3.2 *Mecanismos de sincronização*

Quando são utilizados mecanismos IPC é geralmente necessário utilizar também mecanismos de sincronização da comunicação entre os processos, sendo que os mais frequentemente utilizados são [22]:

- Semáforos;
- Mutex;
- Eventos.

**Semáforos**

Um objecto de semáforo é um objecto de sincronização que mantém uma contagem entre zero e um valor máximo especificado. A contagem decrementa um valor de cada vez que uma *thread* associa a si o semáforo e incrementa um valor de cada vez que a *thread* liberta o semáforo. O estado do semáforo é definido como assinalado quando a contagem é maior que zero e não assinalado quando é zero.

Os semáforos são úteis para controlar um conteúdo partilhado que pode ser acedido por um número limitado de utilizadores. O semáforo actua como uma porta de entrada que limita o número de threads que pode aceder ao conteúdo.

Por exemplo, se uma aplicação cria um determinado número de janelas e utiliza um semáforo para limitar o máximo de janelas aberta simultaneamente. O semáforo decrementa um valor sempre que uma janela é criada e incrementa um valor sempre que uma janela é fechada. Quando a contagem chegar a zero indica que o número limite de janelas abertas foi atingido.

Para criar um objecto de semáforo utiliza-se a função "CreateSemaphore" na qual são especificadas as contagens inicial e máxima. Após a criação do objecto é utilizada a função "CreateThread" para criar as várias threads.

Antes de uma *thread* tentar efectuar uma tarefa ela utiliza a função "WaitForSingleObject" para verificar se a contagem do semáforo permite o acesso ao objecto ou não. Como referido anteriormente, se a função indicar que a contagem é igual a zero retorna de imediato que o semáforo não está assinalado. Caso contrário, a função "WaitForSingleObject" decrementa a contagem do semáforo em 1.

Quando a *thread* completa a tarefa é utilizada a função "ReleaseSemaphore" de modo a incrementar em 1 a contagem do semáforo e permitir que outra *thread* em espera possa executar a sua tarefa.

**Mutex**

Um objecto mutex é um objecto de sincronização semelhante aos semáforos, cujo estado é assinalado quando não está associado a uma *thread* e não assinalado quando está associado a uma *thread*. A principal diferença entre um objecto Mutex e um objecto Semáforo é que apenas uma *thread* pode estar associada simultaneamente a um objecto mutex.

Por exemplo, para prevenir que duas *threads* escrevam na memória partilhada ao mesmo tempo, cada *thread* aguarda que o objecto mutex fique associado a si antes de executar o código que acede à memória. Após escrever na memória partilhada essa *thread* liberta o objecto mutex.

Há semelhança do que se sucede com os semáforos, para criar um objecto de mutex utiliza-se a função "CreateMutex"e a função "CreateThread" para criar as várias *threads*.

Antes de uma *thread* tentar efectuar uma tarefa ela utiliza a função "WaitForSingleObject" para verificar se é possível o acesso ao objecto ou não.

Quando a *thread* completa a tarefa é utilizada a função "ReleaseMutex" de modo a permitir que outra *thread* em espera possa executar a sua tarefa.

**Eventos**

Existem dois tipos de objectos de evento, o de *reset* manual e o de *reset* automático, que se diferenciam pelo modo em que as *threads* passam o estado do objecto para não assinalado.

Um objecto de evento envia um sinal para *thread* a indicar que um determinado evento ocorreu. Quando existe sobreposição de *inputs* e *outputs* a tentarem utilizar o mesmo espaço de memória, o sistema define um objecto de evento específico com o estado assinalado assim que uma das operações sobrepostas tiver sido concluída. Uma *thread* pode especificar diferentes objectos de eventos em várias operações sobrepostas em simultâneo.

Analogamente com o que se sucede com os mecanismos de sincronização vistos anteriormente, para criar um objecto de evento utiliza-se a função "CreateEvent"e a função "CreateThread" para criar as várias *threads*.

Antes de uma *thread* tentar efectuar uma tarefa ela utiliza a função "WaitForSingleObject" para verificar se é possível o acesso ao objecto ou não.

Caso o objecto de evento seja de *reset* manual, é necessário chamar a função "ResetEvent" para passar o estado do objecto para não assinalado. Se for de *reset* automático, assim que a *thread* acabe de aceder à informação pretendida, o estado passa automaticamente para não assinalado.

# 4 Interface de comunicação do gProms com outros programas

O gProms é um sistema de modelação de processos com grande capacidade de simulação, optimização e estimativa de parâmetros (tanto para processos dinâmicos como no estado estacionário) de processos altamente complexos e apresenta várias vantagens relativamente a outras aplicações com o mesmo objectivo.

O gProms utiliza uma linguagem clara e concisa, que permite ao utilizador escrever as equações quase como as mesmas são escritas numa folha. Para além disso os utilizadores podem facilmente colocar comentários de maneira a que outros utilizadores a quem sejam passados os modelos não tenham dificuldades de interpretação dos mesmos.

Outra das principais vantagens do gProms reside no seu poder de modelação. Todos os *solvers* foram projectados para sistemas em grande escala e não existem limites para os mesmos a não ser a memória disponível no computador a ser utilizado.

O gProms pode ser utilizado para uma grande variedade de aplicações, desde a petroquímica, indústria alimentar, farmácia, entre outras. Para além disso, o gProms pode ser utilizado em qualquer processo que possa ser descrito por um modelo matemático. Os algoritmos do gProms, ao contrário de outros simuladores, permitem descrever também processos descontínuos.

O gProms dispõe ainda as suas próprias ferramentas de comunicação com outras aplicações, das quais se destacam a *Foreign Object Interface* (FOI) e a *Foreign Process Interface* (FPI) [25] [26].

## *4.1 Foreign Object Interface*

### 4.1.1 Introdução

A maior parte das relações matemáticas pode ser expressa directamente como equações no gProms. (EQUATION no Modelo do gProms).

No entanto podem existir alguns motivos para serem utilizados *packages* de software externo para expressar algumas relações entre variáveis de um modelo:

- A relação entre variáveis não pode ser expressa em forma algébrica sem introdução de demasiados cálculos intermédios;
- A relação entre variáveis envolve demasiados parâmetros que teriam de ser inseridos manualmente em gProms;
- O software que tem os cálculos necessários já existe e foi testado e não seria produtivo estar a reproduzi-lo integralmente em gProms;

Um exemplo da utilização de *foreign objects* é o cálculo de propriedades físicas, sendo a relação entre a entalpia específica de uma mistura com a sua temperatura, pressão e composição, um dos melhores exemplos da sua aplicação, uma vez que implica uma equação de terceiro grau.

Exprimir este tipo de relações directamente em linguagem de gProms implica a introdução de um grande número de variáveis auxiliares que iriam não só aumentar o tamanho do problema a resolver como fazer com que a convergência fosse mais difícil uma vez que algumas dessas variáveis não têm significado físico directo, pelo que é difícil de obter bons palpites de valores iniciais para eles. Para além disso as equações de estado envolvem parâmetros com pontos críticos para vários componentes da mistura assim como coeficientes de interacção entre eles.

Se existir um *package* de propriedades disponível e testado, o mesmo poderá ser utilizado como *foreign object* com o gProms e usar os seus serviços durante a simulação.

Apesar de a utilização de *foreign object*s ter algumas vantagens, a mesma está sujeita a algumas regras.

Cada simulação de gProms pode interagir com qualquer número de *foreign objects*. Cada *foreign object* fornece um conjunto de métodos que são as rotinas de cálculo que estão acessíveis a aplicações externas.

Por exemplo, um *foreign object* de propriedades físicas iria fornecer os métodos para cálculo das densidades na fase líquida e sólida, entalpias, fugacidades, etc. Por seu lado, um *foreign object* para custos de capital poderia fornecer métodos para calcular os custos do equipamento utilizado.

Um método *foreign object* em gProms calcula um valor (*output*) para determinados valores de uma ou mais variáveis (*inputs*), que pode ser tanto escalares como vectores.

No exemplo atrás referido do cálculo da entalpia específica de uma mistura em fase líquida, a entalpia específica seria o único *output* e a temperatura, pressão e as fracções molares dos componentes da mistura seriam os três *inputs*.

### 4.1.2 Implementação de *foreign objects* nos modelos de gProms

Como referido anteriormente, um *foreign object* fornece um meio de calcular um ou mais valores como funções das variáveis em gProms, pelo que os mesmos logicamente serão descritos no modelo de gProms.

A utilização dos *foreign object*s nos modelos de gProms tem de seguir algumas regras:

**A)** Cada *foreign object* no modelo é declarado como um parâmetro do tipo "FOREIGN OBJECT", seguido do nome do parâmetro entre aspas, uma vez que o mesmo não se encontra definido originalmente como um *input* do gProms. Na figura 4.1 pode-se ver um exemplo da denominação de um *foreign object* (referido como PPP).

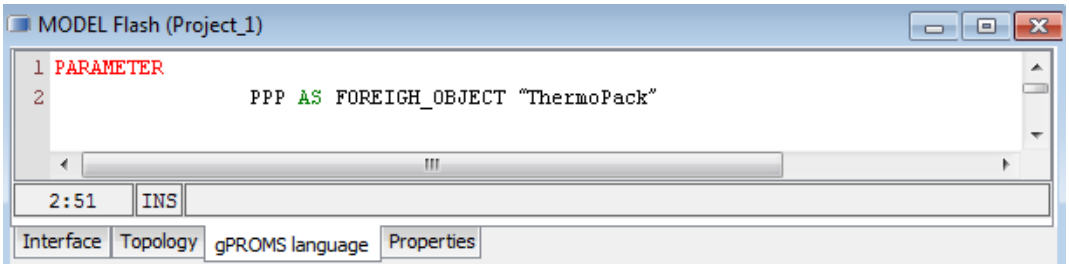


**Figura 4.1 - Exemplo da denominação de um foreign object.**

No exemplo da figura 4.1, o modelo Flash vai utilizar um *foreign object*. Quando necessário, esse *foreign object* vai ser referido como PPP. O PPP é implementado por um software externo (neste caso um *package* com propriedades físicas) chamado "ThermoPack", em que PPP é uma instância do ThermoPack.

**B)** Os métodos de um *foreign object* são referidos como demonstrado abaixo, onde InputList é a lista dos métodos de input.

```
ForeignObjectName.MethodName (InputList)
```

**C)** Cada input de um metido é uma expressão ou variável com valores escalares ou vectoriais.

**D)** Cada método devolve um único valor escalar o vectorial e pode apresentar valores inteiros, reais ou lógicos.

Na figura 4.2 podemos verificar um exemplo de um modelo utilizando um *foreign object* [25].

Na linha 3 é especificado que o modelo Flash utiliza um *foreign object* do tipo ThermoPack, correspondente a uma *package* de um software externo ao gProms. Neste modelo o *foreign object* foi definido como PPP.

Este *foreign object* define vários métodos que devolvem valores de vários tipos que podem ser utilizados em vários locais do modelo. Por exemplo, na linha 13 é utilizado um método chamado NumberOfComponents que retorna um valor inteiro correspondente ao número de compostos presentes na mistura. Este método é utilizado para definir o número de componentes, como o parâmetro NoComp neste modelo.

Na linha 16 são utilizado dois outros métodos do mesmo *foreign object*, LiquidEnthalpy e VapourEnthalpy, que correspondem aos valores de entalpia específica ds fases líquida e gasosa, sendo que cada um deles devolve um valor real, calculado através do *foreign object* a partir dos inputs (T, P, x) e (T, P, y) respectivamente.

Na linha 17 são utilizados mais dois métodos, LiquidFugacityCoeff e VapourFugacityCoeff. Estes métodos devolvem vectores de valores reais.

Na linha 19 é utilizado um método chamado StableLiquid que devolve um valor lógico, indicado se a fase líquida é estável nas condições de temperatura, pressão e composição indicadas.

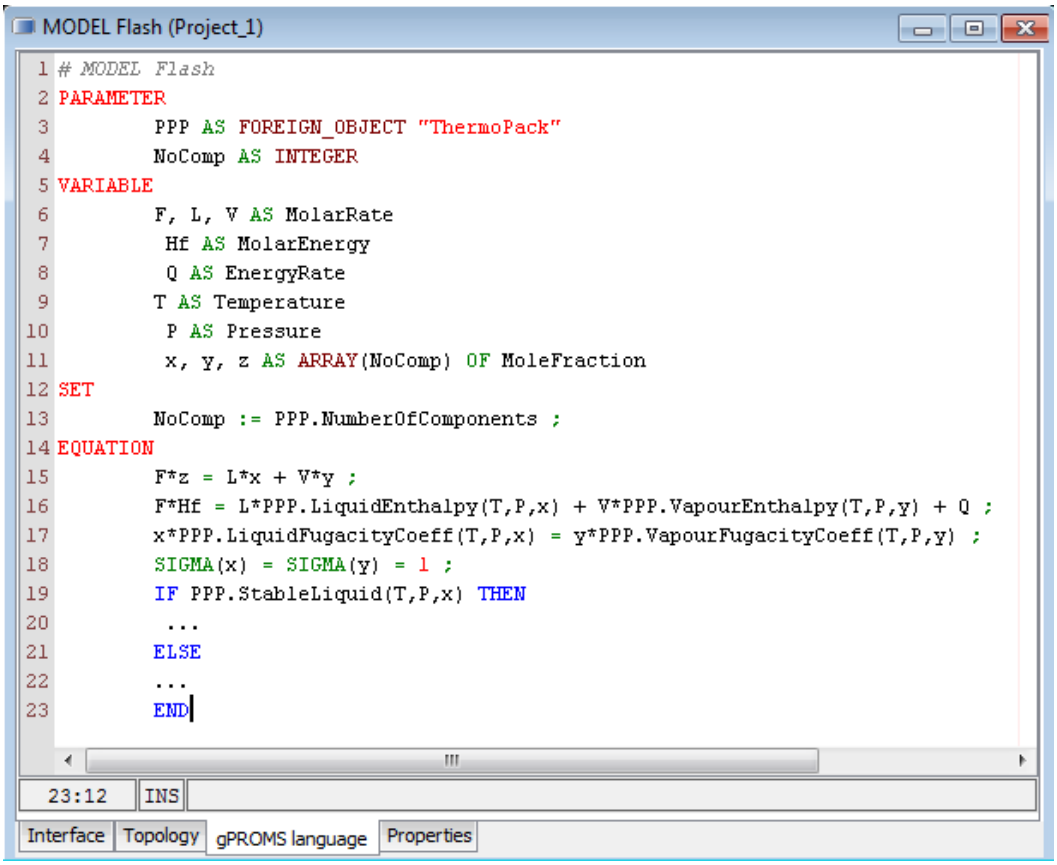


**Figura 4.2 - Exemplo de um modelo utilizando um foreign object.**

### 4.1.3 Procedimentos de inicialização e terminação de um foreign object

Quando é pretendido executar um processo, o gProms constrói uma lista de todos os *foreign objects* que são utilizados por ele e vai tentar criar cada instância dos mesmos. Para tal o gProms tem de chamar as funções de inicialização e terminação do processo.

**Procedimento de inicialização**

A função gFOI é definida por:

```
gFOI (ForeignObjectID, ForeignObjectHandle, Status)
```

Os argumentos da função gFOI encontram-se descritos na tabela 4.1.

**Tabela 4.1 - Argumentos da função gFOI.**

| Nome do argumento | Tipo | Descrição | Local de especificação |
|---|---|---|---|
| ForeignObjectID | Character*256 | Nome completo do foreign object | Entrada |
| ForeignObjectHandle | Integer | Handle para identificar o foreign object sempre que for chamado | Saída |
| Status | Integer | Status da inicialização (Status = 1 significa sucesso) | Saída |

O ForeignObjectID é uma *string* que identifica a instância do *foreign object* que está ser criado.

O Status é a resposta do *foreign object* que indica se o pedido foi efectuado com sucesso e, só em caso positivo, é que é dada continuidade ao processo. Caso não dê positivo (Status diferente de 1) o processo é terminado imediatamente.

Uma vez que o gProms pode utilizar várias vezes o mesmo foreign object, a rotina gFOI devolve ao gProms um ForeignObjectHandle, que é habitualmente um endereço que permite que a instância do *foreign object* seja rapidamente localizada.

**Procedimento de terminação**

A função gFOT é definida por:

```
gFOT (ForeignObjectID, ForeignObjectHandle, Status)
```

Os argumentos da função gFOT são em tudo semelhantes aos da função gFOI, apenas com a diferença que o ForeignObjectHandle é especificado na entrada e mantém-se inalterado na saída.

## *4.2 Foreign Process Interface*

### 4.2.1 Introdução

O *foreign process interface* (FPI) fornece um mecanismo geral para a troca de informação entre as simulações do gProms e software externo.

Esta comunicação ocorre em pontos de tempo discretos durante a duração da simulação, em que o utilizador tem total liberdade em determinar a frequência e o conteúdo das alterações efectuadas.

Uma das principais aplicações do FPI é a utilização das simulações do gProms para estudar e validar o controlo de algoritmos implementados em sistemas externos de controlo em tempo real.

O FPI inclui dois componentes principais:

**A)** O primeiro é o conjunto de tarefas de comunicação. Ao inserir instâncias dessas tarefas no SCHEDULE entra as entidades das opções TASK e PROCESS, o utilizador pode determinar o período de tempo e o conteúdo de qualquer comunicação que ocorra quando o SHEDULE é executado.

**B)** O segundo é o protocolo de comunicação entre o gProms e o software externo. A execução de uma tarefa de comunicação no SCHEDULE automaticamente faz com que o gProms invoque um conjunto de procedimentos solicitados pelo utilizador. O protocolo de comunicação FPH irá especificar qual o procedimento que deverá ser utilizado, o seu nome e a forma precisa dos seus argumentos.

### 4.2.2 Implementação de *foreign process interface* nos modelos de gProms

A simulação dinâmica no gProms é definida nas entidades do PROCESS, que são executadas de modo a correr a simulação.

Cada PROCESS envolve um SCHEDULE de acções (definido pelas entidades da TASK) organizadas em SEQUENCE ou em PARALLEL. Existe também a possibilidade de tarefas de execução condicional (IF) ou iterativas (WHILE).

Uma entidade na TASK pode ser definida em termos de TASK de nível inferior, que por sua vez inclui outras de ainda inferior nível e assim sucessivamente, o que forma uma hierarquia de tarefas.

No nível mais baixo da hierarquia existe um conjunto de tarefas elementares como RESET, REPLACE e CONTINUE.

O FPI introduz cinco novos tarefas elementares que, tal como as outras tarefas elementares, podem ser inseridas num SCHEDULE e a sua função é trocar informação entre um processo a ser executado pelo gProms e um software externo (que a partir deste momento será referido como *foreign process*).

As cinco tarefas de comunicação introduzidas pelo FPI são PAUSE, GET, SEND, SENDMATHINFO e LINEARISE. Todas as tarefas de comunicação são executadas sincronizadamente, o que implica que a simulação é suspensa completamente enquanto o gProms aguarda a resposta do *foreign process*. Nos subcapítulos seguintes iremos abordar as tarefas PAUSE, GET e SEND.

**PAUSE**

A tarefa PAUSE mete a simulação em pausa até que um sinal seja recebido por parte do *foreign process* e a sua sintaxe geral pode ser verificada na figura 4.4.

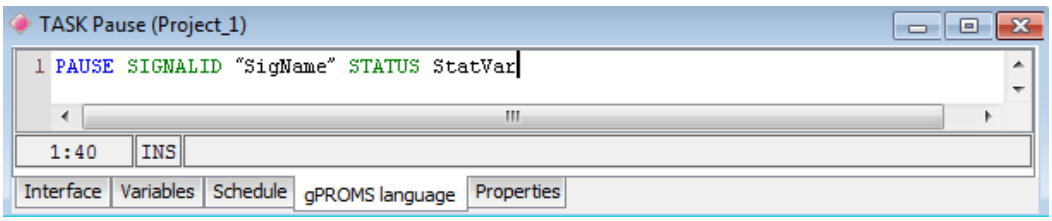


**Figura 4.3 - Sintaxe geral da tarefa PAUSE.**

O “SigName” é uma string de caracteres que são passadas pelo gProms para o *foreign process* quando é executada a tarefa PAUSE e que é utilizada para que este identifique a instância do pedido de PAUSE que está ser solicitada.

O uso do SIGNAL ID é opcional e geralmente só é utilizado quando existem várias tarefas de PAUSE distintas no SCHEDULE.

O “StatVar” é uma variável lógica do gProms que recebe o resultado do pedido de PAUSE ao *foreign process*. Geralmente o valor TRUE indica que existiu sucesso (ou permissão) para continuar a simulação enquanto o valor FALSE sinaliza que não existiu sucesso (ou que existiu um pedido para parar a simulação). O uso do STATUS também é opcional.

Na figura 4.5 pode-se verificar um exemplo de uma tarefa de PAUSE, onde a primeira acção da TASK é utilizar a tarefa PAUSE, enviando o *string* "ExampleStartRequest" para o *foreign process*.

A simulação fica então parada a aguardar que o *foreign process* responda e a execução do resto da TASK depende do valor da variável STATUS que seja respondido pelo *foreign process*, só avançando se for OKtoContinue.

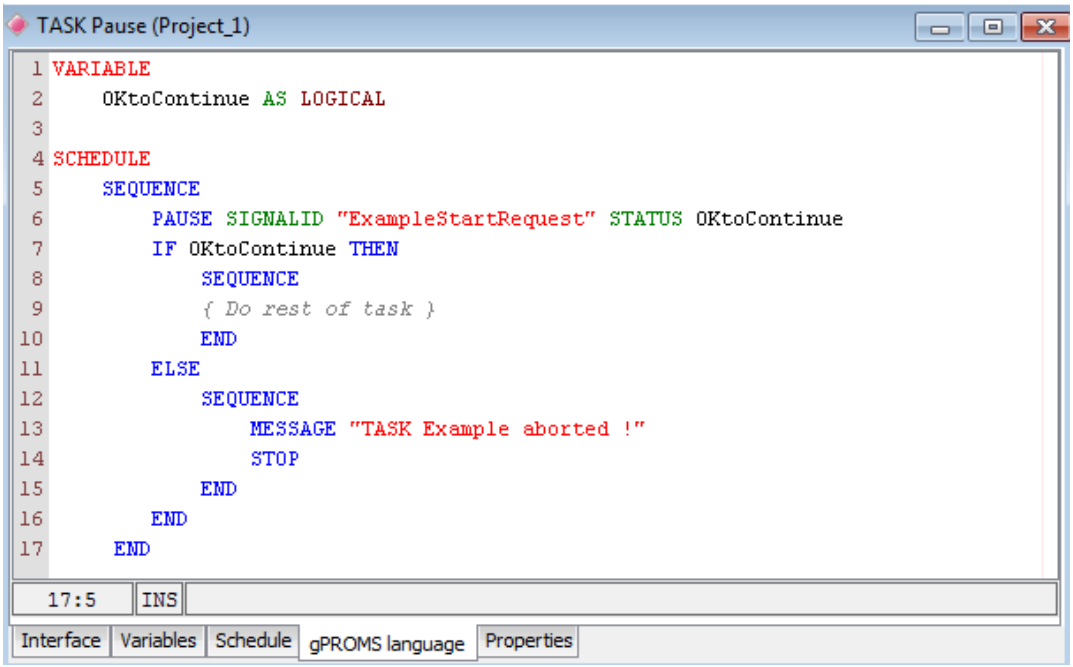


**Figura 4.4 - Exemplo da tarefa PAUSE.**

**GET**

A tarefa GET recebe valores de uma ou mais variáveis do *foreign process* e a sua sintaxe geral pode ser verificada na figura 4.6.

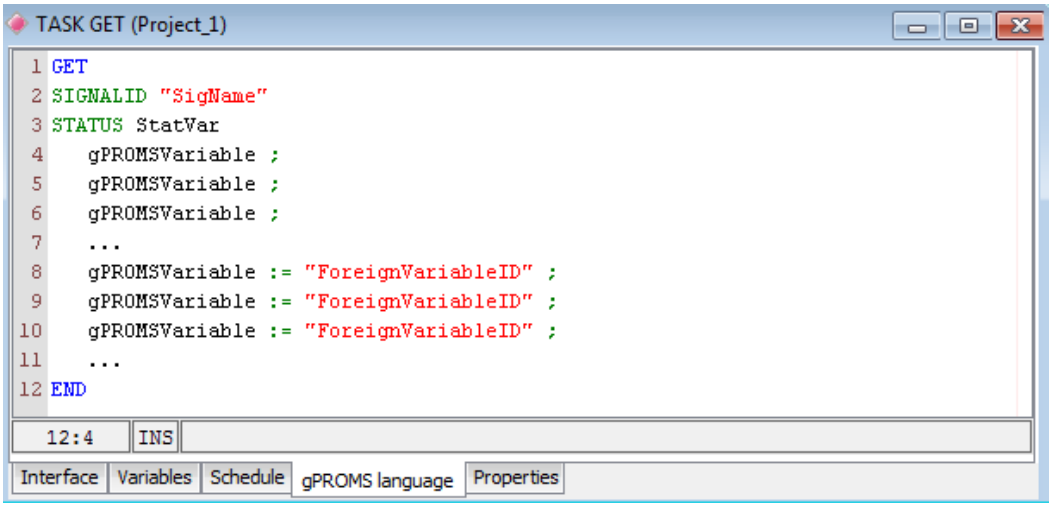


**Figura 4.5 - Sintaxe geral da tarefa GET.**

O SigName é uma *string* de caracteres que será passada pelo gProms para o *foreign process* quando executa a tarefa GET e que pode ser utilizado por este para identificar qual a tarefa GET que está ser solicitada, o que é útil quando existem várias tarefas GET ao longo do SCHEDULE. Em algumas aplicações em que o processo em gProms recebe informação de mais que um *foreign process*, o SigName pode também ser utilizado para identificar a fonte de dados que é pretendida.

O uso do SIGNAL ID é opcional e geralmente só é utilizado quando existem várias tarefas de GET distintas no SCHEDULE.

O "StatVar" é uma variável lógica do gProms que recebe o resultado do pedido de GET ao *foreign process*. Geralmente o valor TRUE indica que existiu sucesso (ou permissão) para continuar a simulação enquanto o valor FALSE sinaliza que não existiu sucesso (ou que existiu um pedido para parar a simulação). O uso do STATUS também é opcional.

A gPROMSVariable pode ser uma variável do modelo em gProms utilizada na simulação ou uma variável local na TASK (do tipo real, inteiro ou lógico).

A ForeignVariableID é o nome pelo qual uma variável é conhecida no *foreign process* e é utilizada para que este identifique qual a informação pretendida pelo gProms de modo a poder retornar essa mesma informação.

Um resultado com sucesso do *foreign process* (StatVar = TRUE) após um pedido de GET fornece à simulação em gProms um novo conjunto de valores para as variáveis que foram pedidas. Estes valores apagam os valores que as variáveis tinham antes da tarefa GET.

**SEND**

A tarefa SEND envia valores de uma ou mais variáveis ao *foreign process* e a sua sintaxe geral pode ser verificada na figura 4.7.

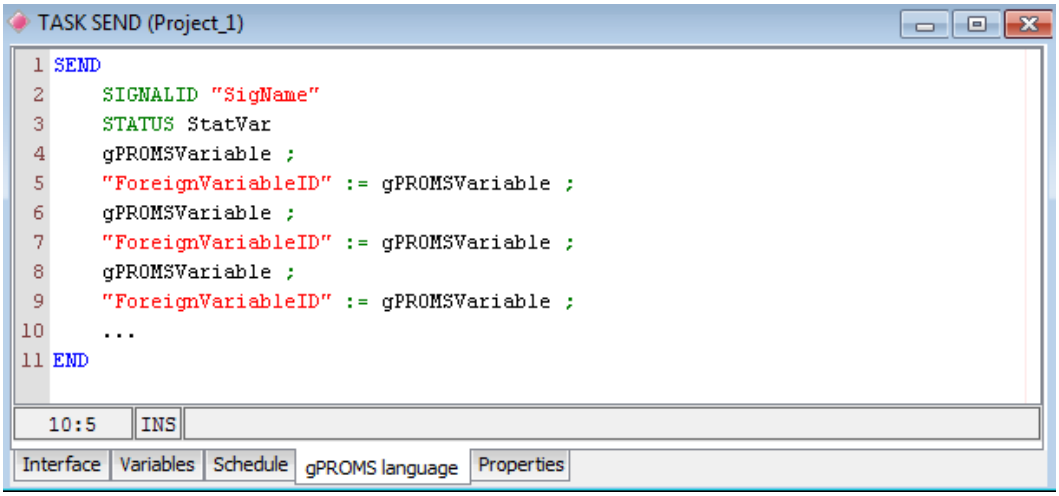


**Figura 4.6 - Sintaxe geral da tarefa SEND.**

O SigName é uma *string* de caracteres que será passada pelo gProms para o *foreign process* quando executa a tarefa SEND e que pode ser utilizado por este para identificar qual a tarefa SEND que está ser solicitada, o que é útil quando existem várias tarefas SEND ao longo do SCHEDULE. Em algumas aplicações em que o processo em gProms envia informação para mais que um *foreign process*, o SigName pode também ser utilizado para identificar para que destino é pretendido o envio.

O uso do SIGNAL ID é opcional e geralmente só é utilizado quando existem várias tarefas de SEND distintas no SCHEDULE.

O "StatVar" é uma variável lógica do gProms que recebe o resultado do pedido de SEND ao *foreign process*. Geralmente o valor TRUE indica que existiu sucesso (ou permissão) para continuar a simulação enquanto o valor FALSE sinaliza que não existiu sucesso (ou que existiu um pedido para parar a simulação). O uso do STATUS também é opcional.

A gPROMSVariable pode ser uma variável do modelo em gProms utilizada na simulação ou uma variável local na TASK (do tipo real, inteiro ou lógico).

A ForeignVariableID é o nome pelo qual uma variável é conhecida no *foreign process* e é utilizada para que este identifique qual a informação pretendida pelo gProms de modo a poder retornar essa mesma informação.

### 4.2.3 Protocolo de comunicação de um foreign process

Como foi verificado no capítulo 4.2.2, ao utilizar as tarefas de comunicação do FPI, o utilizador pode especificar o tempo e a natureza da informação que é trocada com o *foreign process* quando é executada a simulação em gProms.

A comunicação em si é efectuada quando o gProms chama um conjunto de procedimentos. O protocolo de comunicação FPI define a forma de interacção desses procedimentos com o *foreign process*, através da sua lista de argumentos.

Uma implementação do FPI tem de conter cinco procedimentos, gFPPAUSE, gFPGET, gFPSEND, gfPSENDM e gFPLINEARISE, correspondentes às cinco tarefas de comunicação vistas anteriormente, PAUSE, GET, SEND, SENDMATHINFO e LINEARISE.

Para além disso tem ainda de conter dois procedimentos adicionais utilizados pelo processo em gProms para iniciar e terminar a interacção com o *foreign process*, o gFPI e o gFPT respectivamente.

**gFPI**

A função gFPI é utilizada para iniciar a comunicação entre o processo em gProms e o foreign process e é invocada automaticamente com o início do processo em gProms.

A função gFPI é definida por:

```
gFPI (FPID, FPHANDLE, PRNAME, STATUS)
```

Os argumentos da função gFPI encontram-se descritos na tabela 4.2.

**Tabela 4.2 - Argumentos da função gFPI.**

| Argumento | Tipo | Entrada | Saída |
|---|---|---|---|
| FPID | Char*256 | Nome completo do *foreign process* | Não é alterado |
| FPHANDLE | Integer | 0 | Um integer único definido pelo foreign process para identificar as interacções deste processo específico |
| PRNAME | Char*256 | Nome do processo em gProms que iniciou a comunicação | Não é alterado |
| STATUS | Integer | 1 | 1 se sucesso<br>0 se insucesso |

**gFPPAUSE**

Esta função é invocada automaticamente quando uma tarefa PAUSE é executada no SCHEDULE.

A função gFPPAUSE é definida por:

```
gFPPAUSE (FPID, FPHANDLE, PRNAME, SIGNAL, TIME, STATUS)
```

Os argumentos da função gFPPAUSE encontram-se descritos na tabela 4.3.

**Tabela 4.3 - Argumentos da função gFPPAUSE.**

| Argumento | Tipo | Entrada | Saída |
|---|---|---|---|
| FPID | Char*256 | Nome completo do *foreign process* | Não é alterado |
| FPHANDLE | Integer | Identificador associado ao processo em gProms pelo foreign process quando foi efectuado o gFPI | Não é alterado |
| PRNAME | Char*256 | Nome do processo em gProms que iniciou a comunicação | Não é alterado |
| SIGNAL | Char*256 | SigName especificada para a instância da tarefa PAUSE no SCHEDULE | Não é alterado |
| TIME | Real*8 | Tempo na simulação em que a tarefa PAUSE é chamada | Não é alterado |
| STATUS | Integer | 1 | 1 se sucesso<br>0 se insucesso |

**gFPGET**

Esta função é invocada automaticamente quando uma tarefa GET é executada no SCHEDULE. A função gFPGET é definida por:

```
gFPGET (FPID, FPHANDLE, PRNAME, SIGNAL, TIME, N, NAME, ITYPE, X, STATUS)
```

Os argumentos da função gFPGET encontram-se descritos na tabela 4.4.

**Tabela 4.4 - Argumentos da função gFPGET.**

| Argumento | Tipo | Entrada | Saída |
|---|---|---|---|
| FPID | Char*256 | Nome completo do *foreign process* | Não é alterado |
| FPHANDLE | Integer | Identificador associado ao processo em gProms pelo foreign process quando foi efectuado o gFPI | Não é alterado |
| PRNAME | Char*256 | Nome do processo em gProms que iniciou a comunicação | Não é alterado |
| SIGNAL | Char*256 | SigName especificada para a instância da tarefa GET no SCHEDULE | Não é alterado |
| TIME | Real*8 | Tempo na simulação em que a tarefa GET é chamada | Não é alterado |
| N | Integer | Nº de valores a obter do *foreign process* | Não é alterado |
| NAME | Char*256 | Nomes das variáveis a ser obtidas do *foreign process* e especificadas pelo ForeignProcessID da tarefa GET | Não é alterado |
| ITYPE | Integer | Tipo de variável a ser obtida pelo foreign process (1 para Real, 2 para Integer, 3 para Logical) | Não é alterado |
| X | Real*8 | Valores das variáveis a ser obtidas pelo foreign process no momento antes da execução da tarefa GET | Novos valores das variáveis |
| STATUS | Integer | 1 | 1 se sucesso<br>0 se insucesso |

**gFPSEND**

Esta função é invocada automaticamente quando uma tarefa SEND é executada no SCHEDULE. A função gFPSEND é definida por:

```
gFPSEND (FPID, FPHANDLE, PRNAME, SIGNAL, TIME, N, NAME, ITYPE, X, STATUS)
```

Os argumentos da função gFPSEND encontram-se descritos na tabela 4.5.

**Tabela 4.5 - Argumentos do procedimento gFPSEND.**

| Argumento | Tipo | Entrada | Saída |
|---|---|---|---|
| FPID | Char*256 | Nome completo do *foreign process* | Não é alterado |
| FPHANDLE | Integer | Identificador associado ao processo em gProms pelo foreign process quando foi efectuado o gFPI | Não é alterado |
| PRNAME | Char*256 | Nome do processo em gProms que iniciou a comunicação | Não é alterado |
| SIGNAL | Char*256 | SigName especificada para a instância da tarefa SEND no SCHEDULE | Não é alterado |
| TIME | Real*8 | Tempo na simulação em que a tarefa SEND é chamada | Não é alterado |
| N | Integer | Nº de valores a obter do *foreign process* | Não é alterado |
| NAME | Char*256 | Nomes das variáveis a ser obtidas do *foreign process* e especificadas pelo ForeignProcessID da tarefa SEND | Não é alterado |
| ITYPE | Integer | Tipo de variável a ser obtida pelo foreign process (1 para Real, 2 para Integer, 3 para Logical) | Não é alterado |
| X | Real*8 | Valores das variáveis a ser obtidas pelo foreign process no momento antes da execução da tarefa SEND | Não é alterado |
| STATUS | Integer | 1 | 1 se sucesso<br>0 se insucesso |

**gFPT**

A função gFPT é utilizada para terminar a comunicação entre o processo em gProms e o foreign process e é invocada automaticamente com o fim do processo em gProms. A função gFPT é definido por:

```
gFPT(FPID, FPHANDLE, PRNAME, STATUS)
```

Os argumentos da função gFPT encontram-se descritos na tabela 4.6.

**Tabela 4.6 - Argumentos da função gFPT.**

| Argumento | Tipo | Entrada | Saída |
|---|---|---|---|
| FPID | Char*256 | Nome completo do *foreign process* | Não é alterado |
| FPHANDLE | Integer | Identificador associado ao processo em gProms pelo foreign process quando foi efectuado o gFPI | Não é alterado |
| PRNAME | Char*256 | Nome do processo em gProms que iniciou a comunicação | Não é alterado |
| STATUS | Integer | 1 | 1 se sucesso<br>0 se insucesso |

# 5 Optimização utilizando AMPL

O AMPL (acrónimo para **A** **M**athematical **P**rogramming **L**anguage) é uma linguagem de modelação algébrica para optimização de problemas lineares e não lineares, com variáveis discretas ou contínuas [28], tendo sido desenvolvida pelos Bell Laboratories.

O AMPL suporta vários *solvers*, incluindo CPLEX, Gurobi, IPOPT, KNITRO, entre outros, sendo que alguns deles são direccionados para a resolução de determinados tipos de problemas.

Na tabela 5.1 são apresentados alguns dos solvers e os tipos de algoritmo para os quais são utilizados [29].

**Tabela 5.1 - Solvers e tipos de algoritmo para o qual são utilizados.**

| Solver | Tipo de algoritmo |
|---|---|
| CPLEX | Lineares<br>Quadráticos |
| Gurobi | Lineares<br>Quadráticos |
| IPOPT | Não lineares |
| KNITRO | Não lineares |

Uma vez que o problema descrito nesta tese é um não linear, iremos analisar de forma mais detalhada os dois *solvers* atrás referidos que são utilizados para optimizar este tipo de problemas.

## 5.1 IPOPT

O IPOPT (**I**nterior **P**oint **Opt**imizer) é um software para optimização não linear em larga escala e foi desenvolvido em C++ por Andreas Wachter e Carl Laird para encontrar soluções locais de problemas de optimização matemática que sejam descritos na forma indicada abaixo [30].

```
                min       f(x)
x in R^n
s.t.            g_L <= g(x) <= g_U
                x_L <=   x   <= x_U
```

Onde f(x) : R^n -> R é a função objectivo e g(x) : R^n -> R^m são as condições limite. Os vectores g_L e g_U denotam os limites inferiores e superiores das condições limite e os vectores x_L e x_U são os limites inferiores e superiores da variável x.

As funções f(x) e g(x) podem ser não lineares e não convexas mas têm de ser duas vezes continuamente diferenciáveis.

Caso seja pretendido definir uma igualdade nas condições fronteira podem ser escritas ao definir os componentes g_L e g_U com o mesmo valor.

## 5.2 KNITRO

O KNITRO (**N**onlinear **I**nterior point **T**rust **R**egion **O**ptimization) é um software para uso comercial para resolver problemas de optimização matemática em larga escala, tendo sido criado por Richard Waltz, Jorge Nocedal, Todd Plantenga e Richard Byrd, sendo produzido pela Ziena Optimization LLC [31].

Apesar de o KNITRO ser especializado em optimização não linear, também resolve problemas lineares, quadráticos, sistemas de equações não lineares ou uma mistura entre eles.

Os valores a obter nestes problemas têm de ser variáveis contínuas em funções contínuas, podendo estas ser convexas ou não convexas.

O KNITRO dispõe de três algoritmos diferentes de optimização com diferentes características entre eles, de modo a garantir uma maior flexibilidade na resolução dos problemas e permite que mais que um dos algoritmos seja utilizado no processo para a obtenção da solução [32].

O problema deve ser exposto do modo apresentado no exemplo abaixo.

$$\begin{aligned} & \min_{x} && f(x) \\ \text{subject to} \quad & && C_E(x) = 0 \\ & && C_I(x) = 0 \end{aligned}$$

Onde, tal como no IPOPT as funções apresentadas têm de ser duas vezes continuamente diferenciáveis.

# 6 Desenvolvimento e implementação prática da comunicação entre processos

## *6.1 gProms*

De modo a se poder simular o comportamento da coluna de cromatografia no estado transiente, foi descrito o modelo do sistema de duas colunas descrito no capítulo 2.7 no campo "Models" do gProms, sendo que cada ciclo foi separado em 6 passos distintos, representados esquematicamente nas figuras 6.1 a 6.6.

No passo 1 é colocado eluente na coluna 1 e retirado extracto na coluna 2. As restantes válvulas encontram-se fechadas.

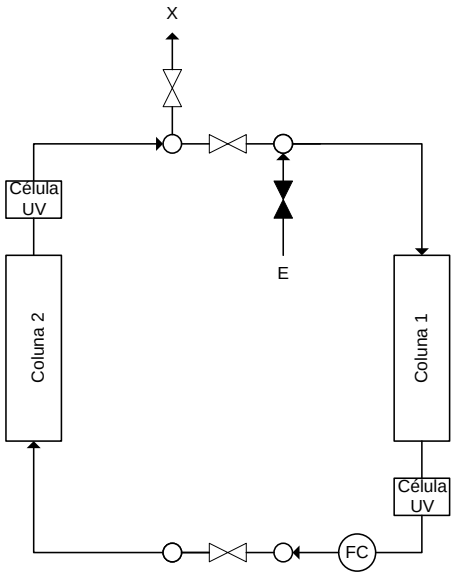


**Figura 6.1 - Passo 1 do ciclo do modelo de duas colunas.**

No passo 2 é colocado eluente na coluna 1, colocada alimentação na coluna 2 e retirado refinado na coluna 2. As restantes válvulas encontram-se fechadas.

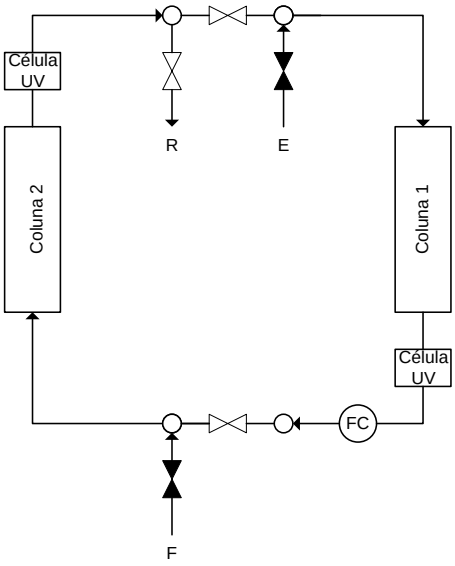


**Figura 6.2 - Passo 2 do ciclo do modelo de duas colunas.**

No passo 3 é colocado eluente na coluna 1 e retirado refinado na coluna 2. As restantes válvulas encontram-se fechadas.

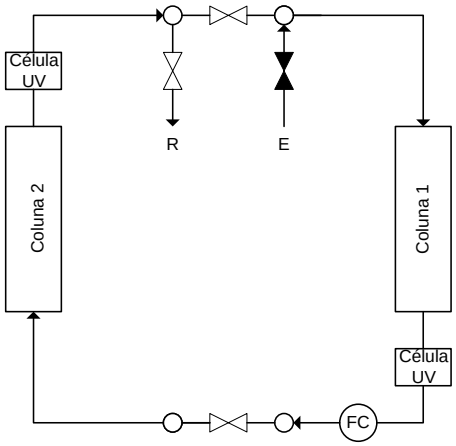


**Figura 6.3 - Passo 3 do ciclo do modelo de duas colunas.**

No passo 4 é colocado eluente na coluna 2 e retirado extracto na coluna 1. As restantes válvulas encontram-se fechadas.

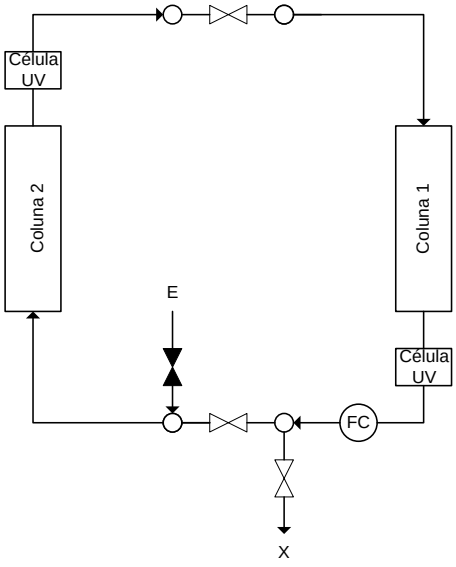


**Figura 6.4 - Passo 4 do ciclo do modelo de duas colunas.**

No passo 5 é colocado eluente na coluna 2, colocada alimentação na coluna 1 e retirado refinado na coluna 1. As restantes válvulas encontram-se fechadas.

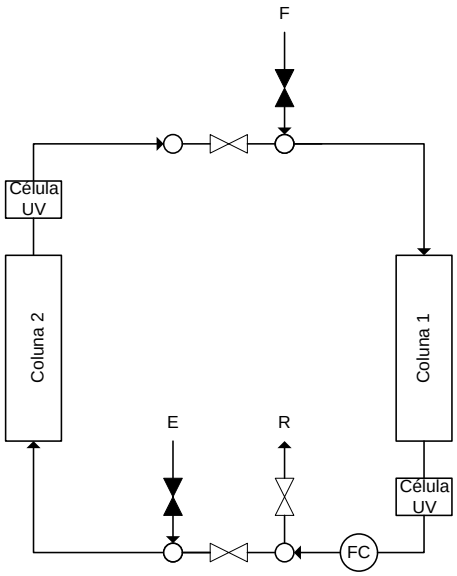


**Figura 6.5 - Passo 5 do ciclo do modelo de duas colunas.**

No passo 6 é colocado eluente na coluna 2 e retirado refinado na coluna 1. As restantes válvulas encontram-se fechadas.

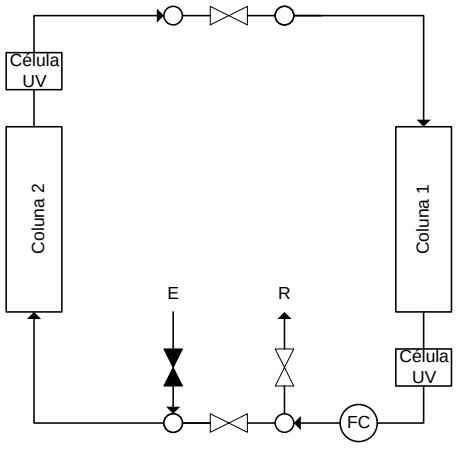


**Figura 6.6 - Passo 6 do ciclo do modelo de duas colunas.**

Como se pode verificar nas figuras 6.1 a 6.6, os passos 1, 2 e 3 são respectivamente simétricos aos passos 4, 5 e 6.

É ainda no campo "Models" que são definidos os parâmetros e as variáveis do modelo, assim como as respectivas equações de balanço dos nós, do modelo da coluna de cromatografia e das purezas do extracto e refinado.

No campo "Processes" são definidos os valores dos parâmetros (utilizando "SET") e das variáveis (utilizando "ASSIGN").

Uma vez que nesta tese o objectivo é utilizar a comunicação entre processo, é também necessária a definição em "SOLUTIONPARAMETERS" da dll que se encontra a ser utilizada.

No campo "Tasks" vai ser definida a interacção do gProms com as outras aplicações, através das tarefas GET, SEND e PAUSE que quando é executada uma simulação, como referido no capítulo 4.2.3, chamam respectivamente as funções gFPGET, gFPSEND e gFPPAUSE.

## 6.2 *Labview/Interface de monitorização, automação e controlo*

Através do Labview foi desenvolvida a interface de monitorização, automação e controlo.

Nas figuras 6.7 a 6.12 encontra-se a interface com os esquemas dos passos 1 a 6 respectivamente do ciclo indicado no capítulo 6.1.

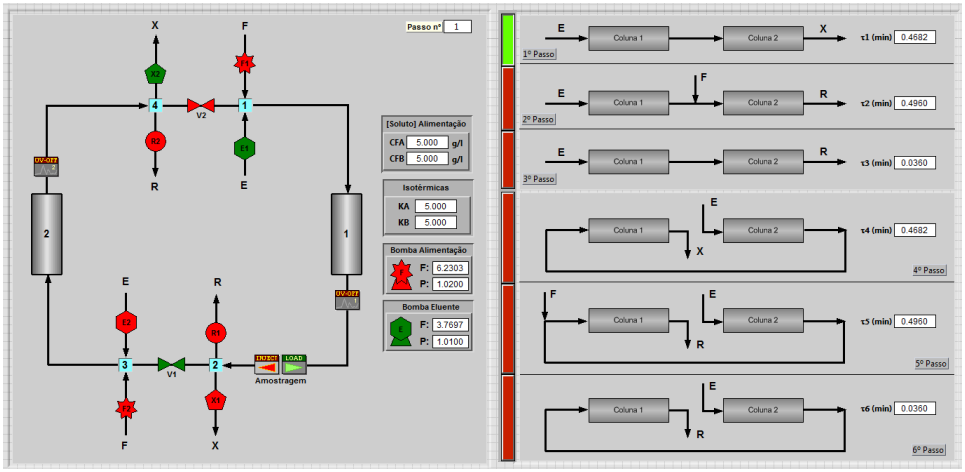


**Figura 6.7 - Interface desenvolvida em Labview com a configuração do passo 1.**

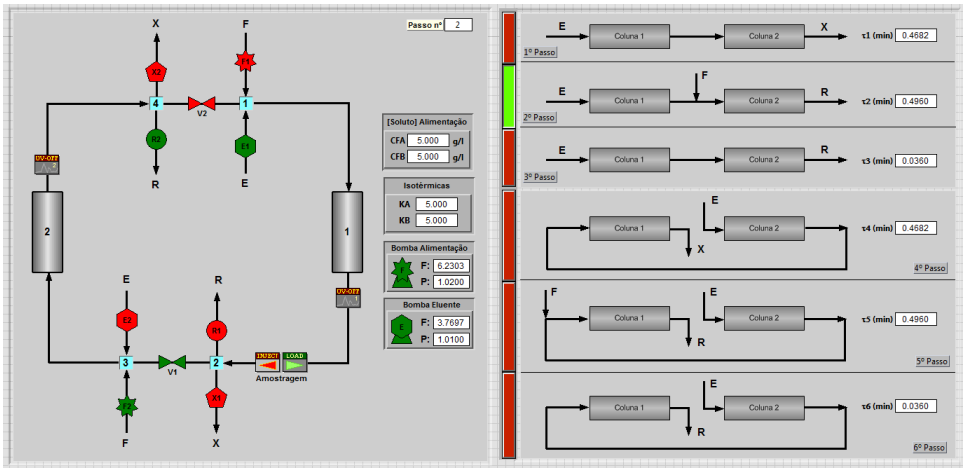


**Figura 6.8 - Interface desenvolvida em Labview com a configuração do passo 2.**

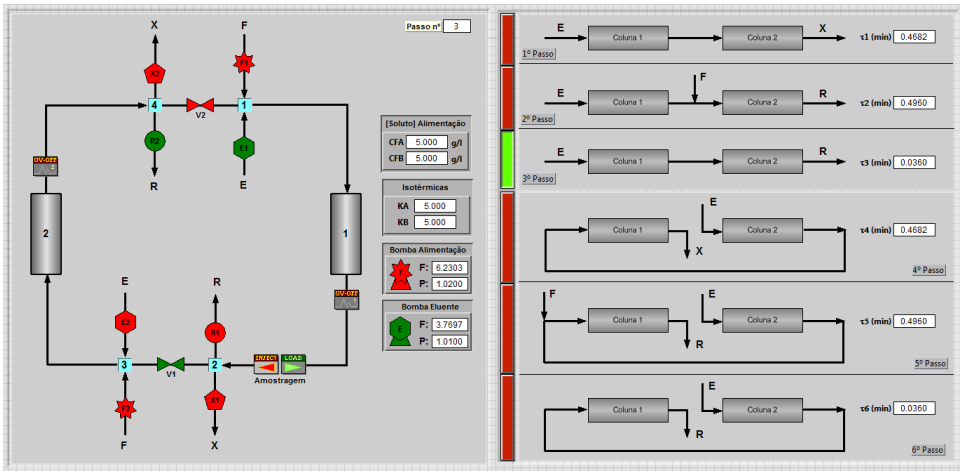


**Figura 6.9 Interface desenvolvida em Labview com a configuração do passo 3.**

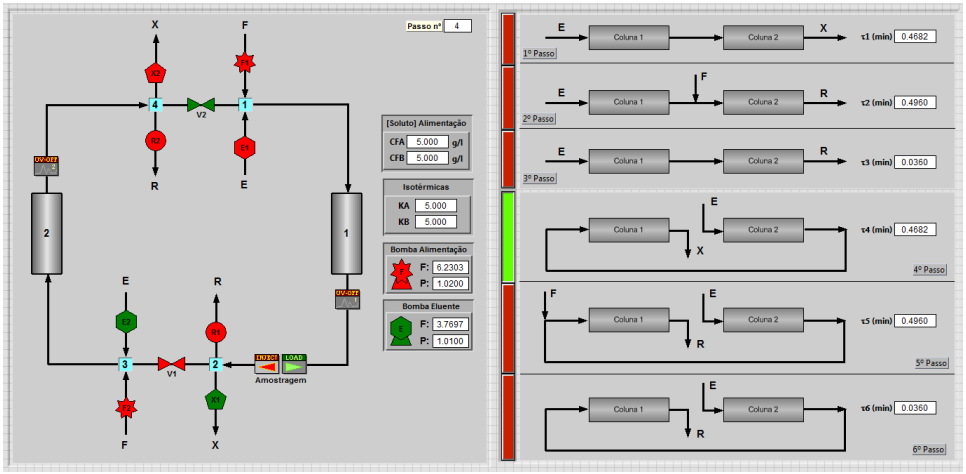


**Figura 6.10 - Interface desenvolvida em Labview com a configuração do passo 4.**

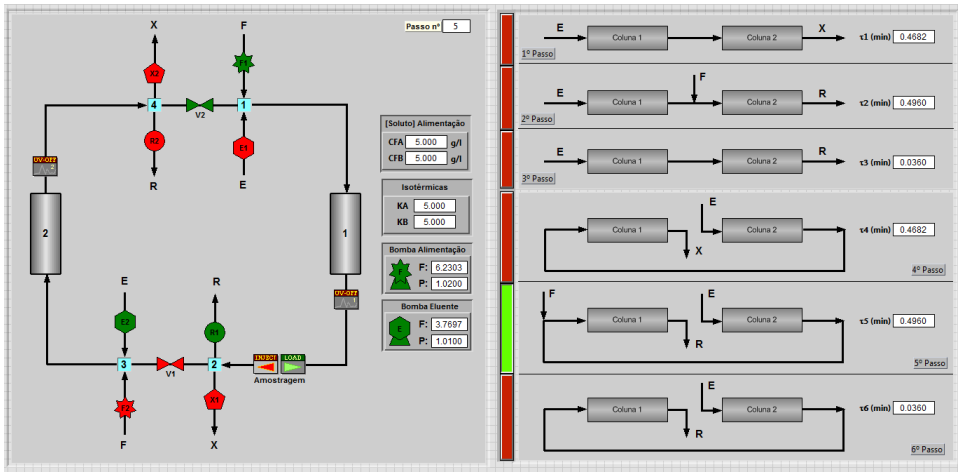


**Figura 6.11 - Interface desenvolvida em Labview com a configuração do passo 5.**

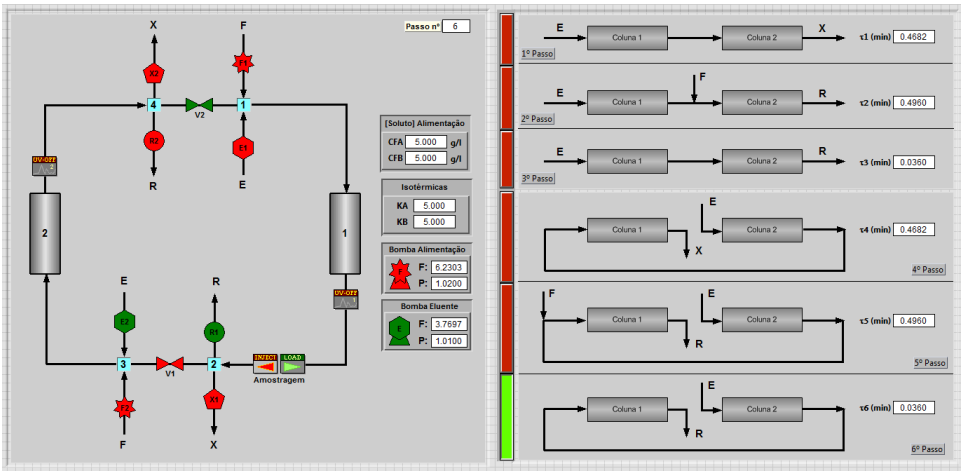


**Figura 6.12 - Interface desenvolvida em Labview com a configuração do passo 6.**

No lado direito das figuras 6.7 a 6.12 estão representados os 6 passos de cada ciclo (sempre visível), sendo ainda neste local da interface de comunicação que pode ser controlado o tempo $\tau$ de cada passo.

Na zona representada no lado esquerdo é feita a alteração do esquema de acordo com o passo em que se encontra. As válvulas e bombas assinaladas a verde significam que se encontram aberta/em funcionamento enquanto as que se encontram a vermelho significam que se encontram fechadas/paradas. É neste local que são controladas as concentrações e as constantes de Henry das isotérmicas de adsorção e em que é possível verificar os caudais da alimentação e eluente que são adicionados ao sistema.

## *6.3 AMPL*

Em AMPL vai ser efectuada a optimização do processo e, tal como acontece em gProms, todo o modelo tem de ser descrito, com a diferença que em AMPL o modelo tem de estar discretizado.

Tal como descrito por Rui Rodrigues [33], a optimização da coluna é efectuada através de um modelo análogo de uma coluna que replica o estado estacionário cíclico da unidade de duas colunas juntamente com uma completa discretização (tanto no espaço como no tempo) [34-35]. Para este efeito a discretização é feita para o ciclo completo (duas unidades de tempo τ) e as condições para o estado estacionário cíclico são directamente impostas [36].

No modelo de optimização utilizado, cujo objectivo é maximizar a produtividade e minimizar o consumo de eluente em relação à alimentação, serão calculados os valores óptimos para as variáveis que mais facilmente podem ser manipuladas numa instalação – os tempos de cada ciclo ($\tau$) e os caudais de alimentação e eluente a inserir no sistema.

A formulação do problema é feita em AMPL e solucionada através de um solver para problemas não lineares (IPOPT).

## *6.4 Visual Studio/C++*

A aplicação Visual Studio é utilizada nesta tese para compilar a dll, desenvolvida através da linguagem C++, que vai ser utilizada na comunicação entre as várias aplicações.

Nesta tese a comunicação entre processos (Labview e gProms) será por File mapping (na Shared Memory) e sincronizada por eventos. São utilizados dois buffers, um para a obtenção por parte do gProms dos valores das variáveis obtidos a partir de Labview e outro para o envio dos resultados obtidos em gProms.

As variáveis estudadas nesta tese serão as concentrações dos caudais de entrada da alimentação, as constantes de Henry das isotérmicas de adsorção e os tempos de *switch* de cada passo do ciclo. Os resultados obtidos que vão ser transferidos para a Shared Memory serão as concentrações de saída de cada soluto em cada coluna ao longo do tempo e ao longo das colunas (caracterização temporal e axial).

Como é possível verificar na figura 6.13, através da informação que é passada por uma aplicação para a Shared Memory (memória partilhada), qualquer outra aplicação (ou instância da mesma aplicação) consegue aceder a essa mesma informação e fazer troca de dados.

Apenas uma aplicação necessita de inicializar efectivamente este processo (as restantes aplicações também utilizam as suas funções de inicialização mas sem efeito prático) mas para que o processo seja terminado todas as aplicações têm de ter sido terminadas, ou seja, basta que uma aplicação se mantenha a correr para que a informação se mantenha na Shared Memory.

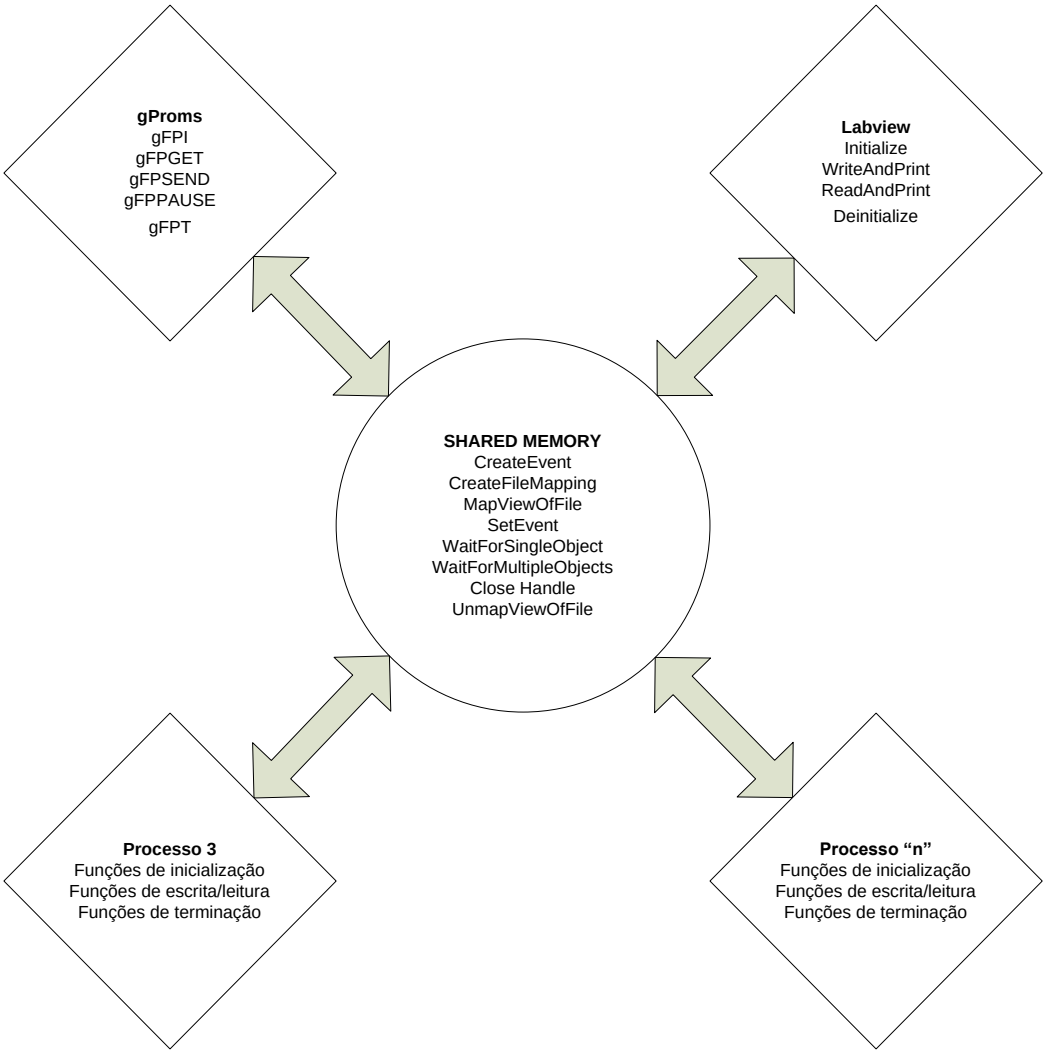


**Figura 6.13 - Representação esquemática da comunicação entre processos.**

## *6.5 Implementação*

### 6.5.1 Fase de testes

Na fase inicial do desenvolvimento e da implementação do protocolo de comunicação entre os processos, foram efectuados vários testes para confirmação de que a comunicação se encontrava a ser efectuada com sucesso.

Numa primeira fase foi testada a comunicação através de Shared Memory entre várias aplicações – Labview, gProms e uma aplicação DOS desenvolvida por Swarajya Pendharkar.

Nesta fase o objectivo era testar a comunicação entre os vários processos escrevendo um único valor que pudesse ser lido pelas diversas aplicações.

Na figura 6.14 encontra-se um exemplo em que uma instância da aplicação DOS escreveu na Shared Memory e em que uma outra instância da mesma aplicação, o Labview e o gProms leram dessa mesma posição de memória onde foi escrita a informação.

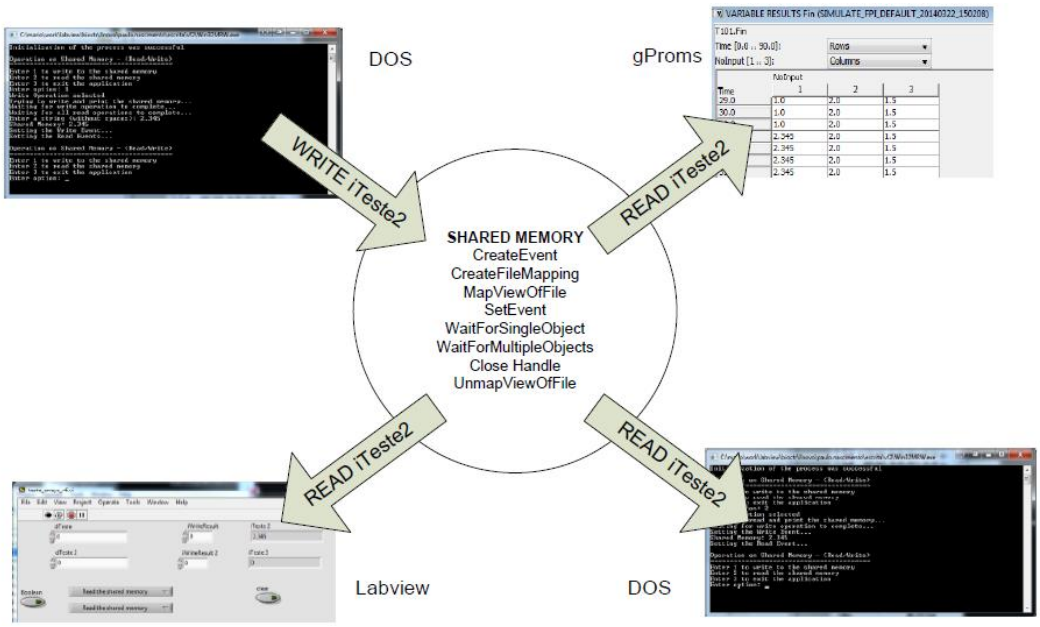


**Figura 6.14 - Comunicação na fase de testes de um único valor entre várias aplicações utilizando Shared Memory.**

Neste caso a primeira instância da aplicação DOS escreveu na Shared Memory o valor "2.345". O Labview através de sua função de leitura (ReadAndPrint) obteve esse mesmo valor a partir da Shared Memory (campo iTeste2), uma segunda instância da aplicação DOS obteve o mesmo valor da Shared Memory, através da sua função de leitura e uma simulação do gProms que utilizou a tarefa GET para obter o valor através da Shared Memory para uma variável (neste caso para um caudal de entrada).

Numa fase posterior foi testado a comunicação através do envio de vectores com vários valores entre as aplicações, como pode ser verificado na figura 6.15.

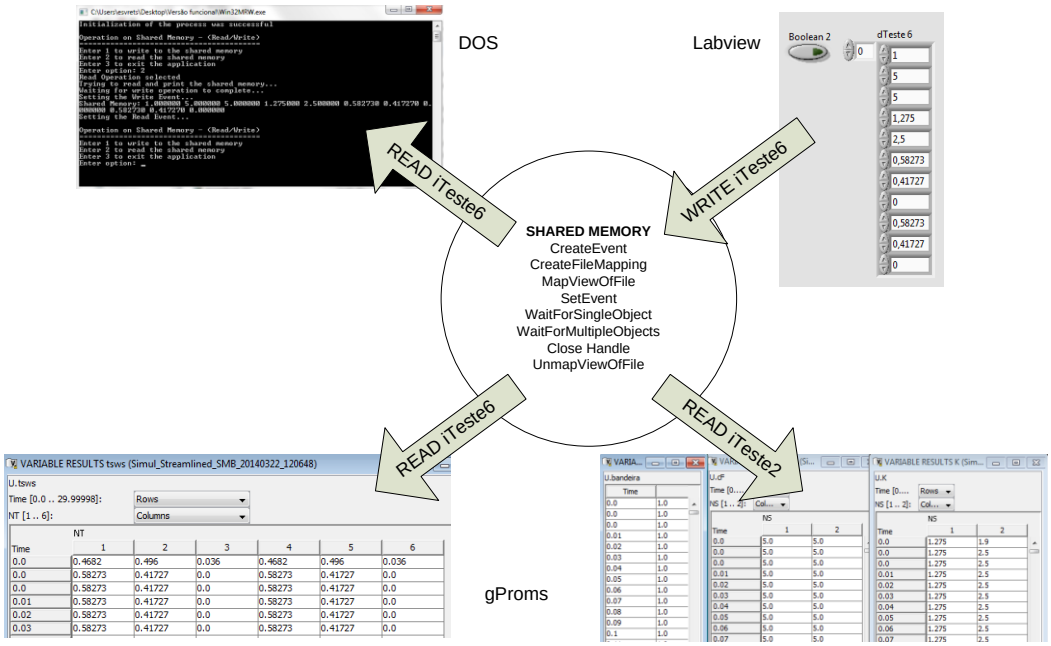


**Figura 6.15 - Comunicação na fase de testes de um vector entre várias aplicações utilizando Shared Memory.**

Neste caso o Labview através da sua função de escrita (WriteAndPrint) enviou um vector com 11 valores. A aplicação DOS através da sua função de leitura obteve os mesmos 11 valores através da Shared Memory (na mesma linha, separados por um espaço, de acordo com o código desenvolvido pelo seu criador), enquanto o gProms através da função GET obtém os mesmos valores da Shared Memory para as suas 11 variáveis (neste caso as variáveis bandeira, cF1, cF2, K1, K2, tsws1, tsws2, tsws3, tsws4, tsws5 e tsws6 respectivamente).

### *6.5.2* Simulação e optimização

Foram efectuadas várias simulações de condições de operação do modelo de duas colunas utilizado nesta tese.

Inicialmente foi feita a simulação com as condições padrão utilizadas nesta tese (simulação #1 na tabela 6.1), sendo posteriormente comparados os resultados com resultados de simulações efectuadas alterando algumas das condições.

Na simulação #2 foi diminuída a concentração de A na alimentação, na simulação #3 foi aumentada essa mesma concentração, na simulação #4 foi feita a aproximação dos

valores da constante das isotérmicas e na simulação #5 foi feito afastamento dessa mesma constante, tendo sido atribuídos os valores indicados na tabela 6.1.

**Tabela 6.1 - Simulações efectuadas em gProms com variação das condições de operação da coluna.**

| # Simulação | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $Q_E$ [ml/min] | 6,23031 | 6,23031 | 6,23031 | 6,23031 | 6,23031 |
| $Q_F$ [ml/min] | 3,76969 | 3,76969 | 3,76969 | 3,76969 | 3,76969 |
| cF(A) [g/L] | 5 | 3 | 7 | 5 | 5 |
| cF(B) [g/L] | 5 | 5 | 5 | 5 | 5 |
| K(A) | 1,275 | 1,275 | 1,275 | 1,275 | 1,275 |
| K(B) | 1,9 | 1,9 | 1,9 | 1,5 | 2,5 |
| $\tau$ (passo 1) | 0,4682 | 0,4682 | 0,4682 | 0,4682 | 0,4682 |
| $\tau$ (passo 2) | 0,4960 | 0,4960 | 0,4960 | 0,4960 | 0,4960 |
| $\tau$ (passo 3) | 0,0360 | 0,0360 | 0,0360 | 0,0360 | 0,0360 |
| $\tau$ (passo 4) | 0,4682 | 0,4682 | 0,4682 | 0,4682 | 0,4682 |
| $\tau$ (passo 5) | 0,4960 | 0,4960 | 0,4960 | 0,4960 | 0,4960 |
| $\tau$ (passo 6 | 0,0360 | 0,0360 | 0,0360 | 0,0360 | 0,0360 |
| $Q_E/Q_F$ | 1,65 | 1,65 | 1,65 | 1,65 | 1,65 |
| Pureza X (%) | 99,0 | 98,2 | 99,2 | 62,4 | 73,2 |
| Pureza R (%) | 99,0 | 98,1 | 95,6 | 92,8 | 95,2 |

Como se pode verificar nos resultados obtidos, a simples alteração de uma das condições de funcionamento da coluna faz com que existam alterações significativas na produtividade da coluna. Nestas simulações foi sempre mantido o mesmo caudal de alimentação e eluente.

Foram posteriormente efetuadas optimizações em AMPL, tendo como objectivo 99% de pureza tanto do refinado como do extracto e a minimização da quantidade de eluente em relação à quantidade de alimentação, garantindo sempre a pureza necessária.

Nas tabelas 6.2 e 6.3 encontram-se os resultados das simulações efectuadas utilizando os dados obtidos pelas optimizações, alterando não só os tempos de cada passo (numa simulação intermédia) e também os caudais de alimentação e eluente.

**Tabela 6.2 - Simulações efectuadas em gProms antes e após optimização em AMPL com alteração de concentração de alimentação.**

| # Simulação | 2 | 6 | 7 | 3 | 8 | 9 |
|---|---|---|---|---|---|---|
| $Q_E$ [ml/min] | 6,23031 | 6,23031 | 6,17916 | 6,23031 | 6,23031 | 6,25899 |
| $Q_F$ [ml/min] | 3,76969 | 3,76969 | 3,82084 | 3,76969 | 3,76969 | 3,74101 |
| cF(A) [g/L] | 3 | 3 | 3 | 7 | 7 | 7 |
| cF(B) [g/L] | 5 | 5 | 5 | 5 | 5 | 5 |
| K(A) | 1,275 | 1,275 | 1,275 | 1,275 | 1,275 | 1,275 |
| K(B) | 1,9 | 1,9 | 1,9 | 1,9 | 1,9 | 1,9 |
| $\tau$ (passo 1) | 0,4682 | 0,4982 | 0,4982 | 0,4682 | 0,45212 | 0,45212 |
| $\tau$ (passo 2) | 0,4960 | 0,4892 | 0,4892 | 0,4960 | 0,50106 | 0,50106 |
| $\tau$ (passo 3) | 0,0360 | 0,0180 | 0,0180 | 0,0360 | 0,04682 | 0,04682 |
| $\tau$ (passo 4) | 0,4682 | 0,4982 | 0,4982 | 0,4682 | 0,45212 | 0,45212 |
| $\tau$ (passo 5) | 0,4960 | 0,4892 | 0,4892 | 0,4960 | 0,50106 | 0,50106 |
| $\tau$ (passo 6 | 0,0360 | 0,0180 | 0,0180 | 0,0360 | 0,04682 | 0,04682 |
| $Q_E/Q_F$ | 1,65 | 1,65 | 1,64 | 1,65 | 1,65 | 1,67 |
| Pureza X (%) | 98,2 | 99,2 | 99,0 | 99,2 | 99,1 | 99,0 |
| Pureza R (%) | 98,1 | 99,1 | 99,0 | 95,6 | 98,3 | 99,0 |

**Tabela 6.3 - Simulações efectuadas em gProms antes e após optimização em AMPL com alteração dos valores da constante de Henry.**

| # Simulação | 4 | 10 | 11 | 5 | 12 | 13 |
|---|---|---|---|---|---|---|
| $Q_E$ [ml/min] | 6.23031 | 6.23031 | 8,05470 | 6.23031 | 6.23031 | 4,94141 |
| $Q_F$ [ml/min] | 3.76969 | 3.76969 | 1,94530 | 3.76969 | 3.76969 | 5,05859 |
| cF(A) [g/L] | 5 | 5 | 5 | 5 | 5 | 5 |
| cF(B) [g/L] | 5 | 5 | 5 | 5 | 5 | 5 |
| K(A) | 1,275 | 1,275 | 1,275 | 1,275 | 1,275 | 1,275 |
| K(B) | 1,5 | 1,5 | 1,5 | 2.5 | 2.5 | 2.5 |
| τ (passo 1) | 0,4682 | 0,34829 | 0,34829 | 0,4682 | 0,58273 | 0,58273 |
| τ (passo 2) | 0,4960 | 0,41033 | 0,41033 | 0,4960 | 0,41727 | 0,41727 |
| τ (passo 3) | 0,0360 | 0,24138 | 0,24138 | 0,0360 | 0 | 0 |
| τ (passo 4) | 0,4682 | 0,34829 | 0,34829 | 0,4682 | 0,58273 | 0,58273 |
| τ (passo 5) | 0,4960 | 0,41033 | 0,41033 | 0,4960 | 0,41727 | 0,41727 |
| τ (passo 6 | 0,0360 | 0,24138 | 0,24138 | 0,0360 | 0 | 0 |
| $Q_E/Q_F$ | 1,65 | 1,65 | 4,14 | 1,65 | 1,65 | 0,98 |
| Pureza X (%) | 62,4 | 55,6 | 99,0 | 73,2 | 72,6 | 99,0 |
| Pureza R (%) | 92.8 | 87,8 | 99,0 | 95,2 | 99,9 | 99,9 |

### 6.5.3 Visualização dos resultados

Após se efectuar as simulações e optimizações com as condições especificadas na interface, os resultados são apresentados na mesma.

Na figura 6.16 encontra-se representado o resultado obtido para o perfil de concentração dos solutos A e B nas colunas 1 e 2, ao longo do tempo e onde é facilmente distinguido em que período de tempo τ se encontra cada zona do perfil.

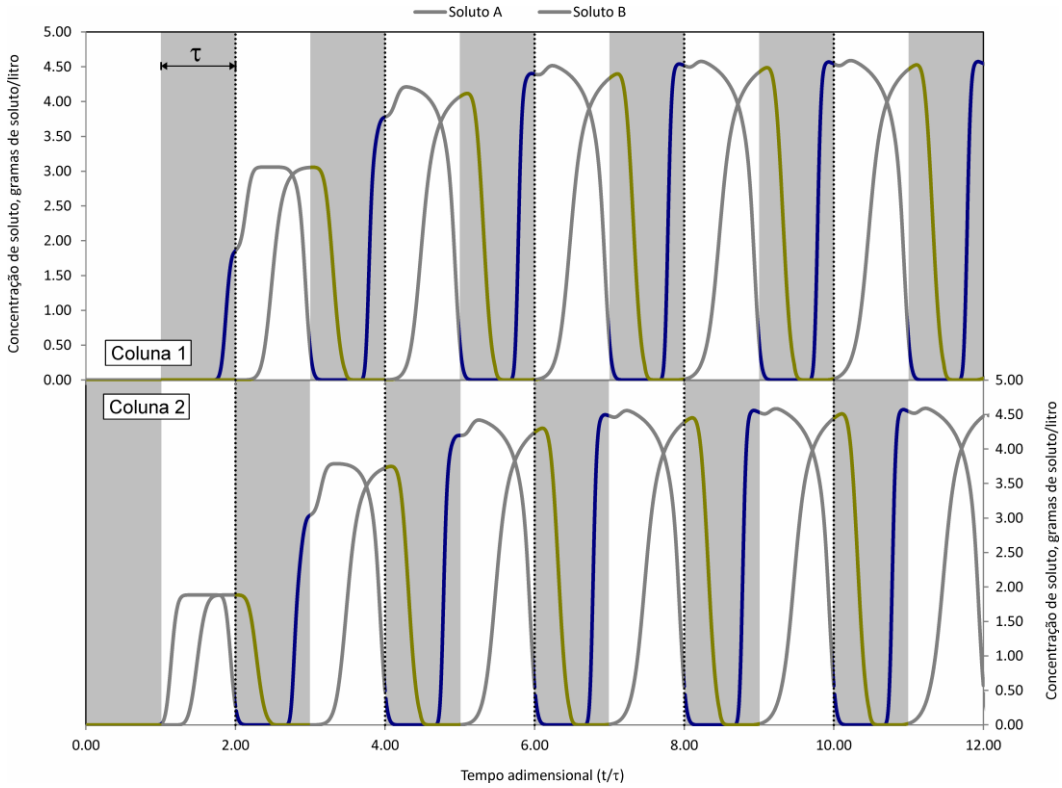


**Figura 6.16 - Perfil de concentração dos solutos A e B ao longo do tempo nas colunas 1 e 2.**

Através da figura 6.16 é possível fazer um *zoom in* de modo a se poder visualizar mais detalhadamente o perfil de concentração ao longo de um ciclo após o processo atingir o estado estacionário cíclico.

Como se pode verificar na figura 6.17, ao se fazer *zoom in* tem-se acesso ao perfil de um ciclo, dividido pelos 6 passos do modelo utilizado nesta tese e onde se pode verificar mais claramente se a recolha de extracto e refinado é feita em locais onde só existe esse soluto ou se o processo poderia ser optimizado.

No caso concreto da figura 6.17 é possível ao utilizador verificar que poderia aumentar o tempo do 1º passo (e consequentemente do 4º passo) de modo a obter uma maior pureza na recolha do soluto A e aumentar o tempo do 3º passo (e 6º passo) para obter também uma maior pureza na recolha do soluto B.

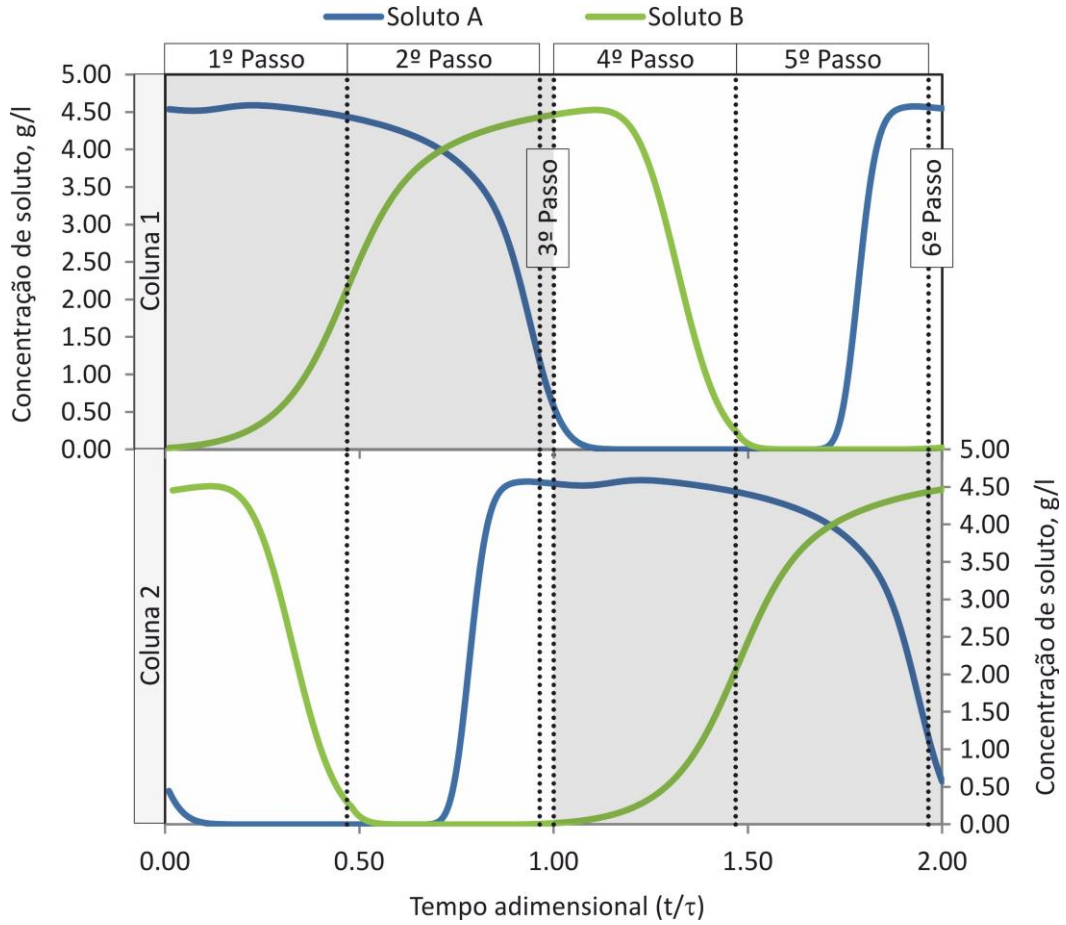


**Figura 6.17 - Perfil de concentração dos solutos A e B nas colunas 1 e 2, ao longo do tempo e num ciclo do processo.**

Das simulações efetuadas e detalhadas nas tabelas 6.1 a 6.3, as que se verificou que tinham mais impacto na produtividade do sistema foram aquelas em que existia alteração no K das isotérmicas.

Na figura 6.18 encontra-se representado o perfil de concentração dos solutos A e B numa das colunas (uma vez que são cíclicas a outra coluna apresentará o mesmo perfil a cada unidade de tempo τ) na situação em que os valores de K eram próximos um do outro. Como se pode verificar pela imagem esta tratava-se de um separação bastante difícil de efectuar e que a interface iria permitir correr várias simulações de modo a obter as condições óptimas de funcionamento.

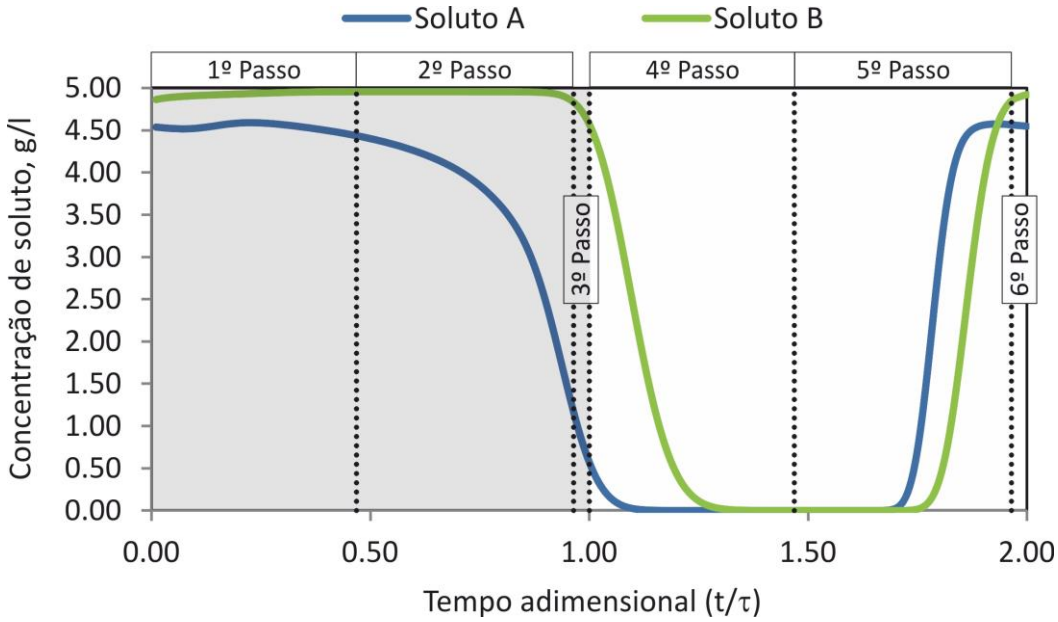


**Figura 6.18 - Perfil de concentração dos solutos A e B numa coluna ao longo do tempo.**

No caso em que os valores de K eram afastados um do outro, cujo perfil de concentrações se encontra na figura 6.19, verificou-se que, ao utilizar os tempos de cada passo representados na figura, não se estava a maximizar a produtividade do sistema.

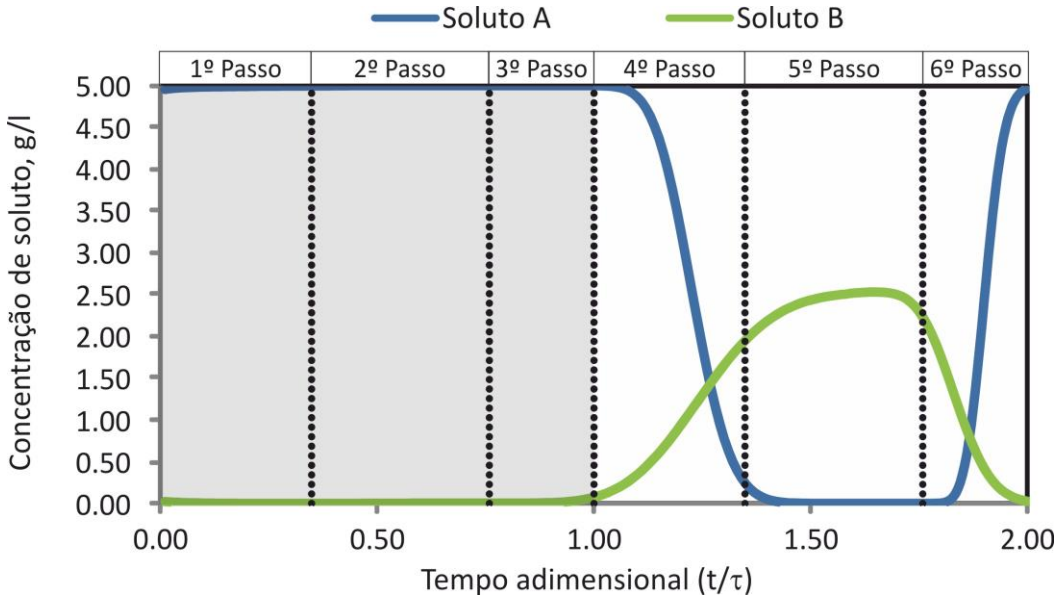


**Figura 6.19 - Perfil de concentração dos solutos A e B ao longo da coluna num caso em que o K de ambos é bastante afastado.**

Nesse sentido recorreu-se a uma optimização em AMPL, cujo resultado se encontra na figura 6.20, onde se pode verificar que com a alteração dos tempos de cada passo (ainda que ligeira) e com alterações em termos de caudais de eluente e de alimentação que se conseguiu uma maior produtividade, fazendo com que as recolhas sejam efetuadas nos pontos correctos.

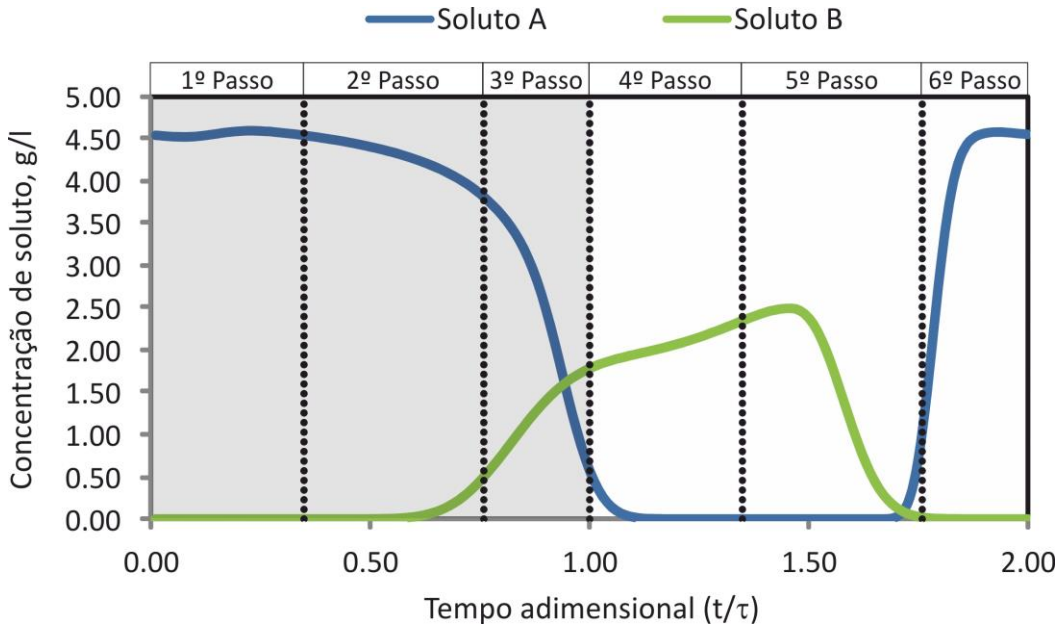


**Figura 6.20 - Perfil de concentração dos solutos A e B ao longo da coluna num caso em que o K de ambos é bastante afastado, após optimização em AMPL.**

Por fim é também possível fazer a comparação directa das simulações em gProms e das optimizações em AMPL, utilizando a mesma interface de monitorização, automação e controlo.

Tendo em conta o perfil de concentrações verificado na figura 6.17 e sobrepondo o perfil de concentração para as mesmas condições mas optimizado através de AMPL, podemos verificar na figura 6.21 que o modelo que nos encontrávamos a utilizar já se encontrava bastante próximo do óptimo.

De referir que os algoritmos utilizados pelo gProms são diferentes dos utilizados pelo AMPL, pelo que o mesmo modelo pode apresentar algumas diferenças nos perfis de concentração, tal como se verifica na figura 6.21.

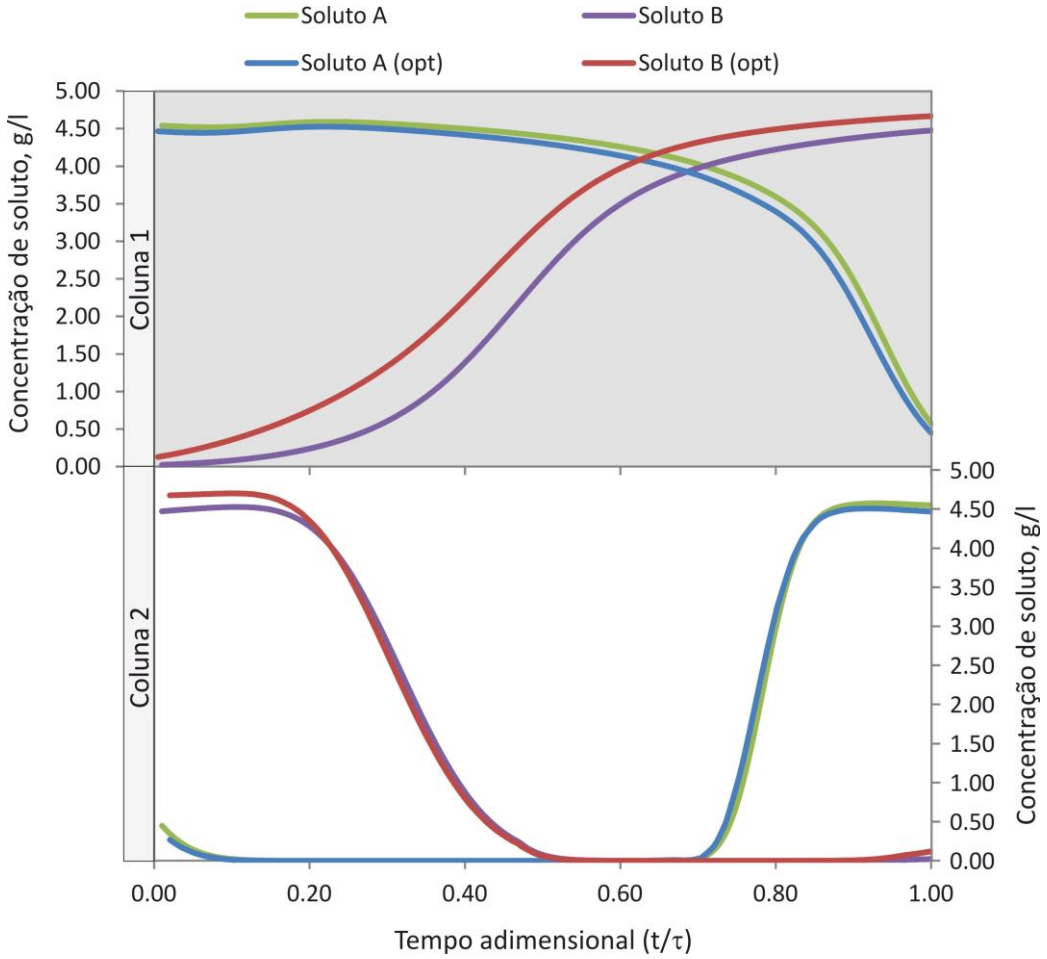


**Figura 6.21 - Perfis de concentração dos solutos A e B numa coluna obtidos através de simulação em gProms e optimização em AMPL.**

# 7 Discussão e conclusões

A comunicação efectuada entre a interface de monitorização, automação e controlo desenvolvida em Labview com gProms e AMPL mostrou-se bastante importante para conseguir visualizar os perfis de concentração ao longo do tempo das simulações efectuadas, permitindo prever o comportamento da coluna de cromatografia.

Ao existir a comunicação entre a interface de monitorização, automação e controlo, desenvolvida em Labview, e os resultados de simulação de gProms, é possível verificar se as condições óptimas se encontram a ser utilizadas, comparando os perfis de concentração reais com os perfis obtidos pelas simulações em gProms. Caso se verifiquem divergências o utilizador tem conhecimento de que as condições de operação foram alteradas.

Efectuando a comunicação da interface com o AMPL podemos achar as novas condições óptimas de funcionamento da coluna.

Como verificado no capítulo 6.6, a variação das condições de funcionamento da coluna de cromatografia (isotérmicas, concentrações e tempos de cada passo) tem um impacto significativo na pureza dos produtos a recolher e também na quantidade de eluente que é necessário utilizar, pelo que se torna bastante importante conseguir obter os resultados de simulações e optimização acima referidos.

Analisando as simulações efectuadas verificou-se que quando os valores de K de ambos os solutos é bastante próximo entre si que a separação é bastante difícil. Através das simulações efectuadas em gProms e da optimização em AMPL, verificou-se que apesar de a separação ser possível é necessária grande quantidade de eluente e que as quantidades recolhidas são bastante pequenas.

Por outro lado se o K de cada soluto for bastante diferente a separação é bastante fácil pelo que a utilização de um processo SMB pode não ser necessária. No caso que foi simulado nesta tese verificou-se que passou de 3 para 2 passos necessários.

Tendo um utilizador acesso a esta informação através da interface, pode desde logo verificar as alternativas para efectuar a separação pretendida, desde a alteração de outras variáveis do processo, mudar a configuração ou adicionar colunas ou até utilizar outro método, provando mais uma vez a grande utilidade deste tipo de interfaces de comunicação com outras aplicações.

Como verificado nas simulações efetuadas, ter acesso ao perfil de concentrações pode não ser o suficiente para garantir a maior produtividade possível e o menor consumo de eluente, pois o facto de ajustar os tempos de cada passo não resulta necessariamente e directamente no atingir desses objectivos. Ao ter acesso simultaneamente a dados de simulação e optimização através de uma interface única consegue-se mais facilmente determinar o que é necessário alterar para se atingir as condições óptimas de operação.

A possibilidade de usar o gProms como ferramenta de simulação dinâmica e o AMPL como ferramenta de optimização a partir da interface de monitorização, automação e controlo permitirá, no futuro, o desenvolvimento e aplicação de estratégias de controlo em linha ao SMB, de forma a optimizar o seu funcionamento quando as concentrações de alimentação, as isotérmicas ou as temperaturas de operação sofrem alterações.

Em suma, a utilização de uma interface de monitorização, automação e controlo que comunique com outras aplicações específicas para determinadas funções trás grandes vantagens ao utilizador, pois permite não só ter acesso a dados em tempo real do seu processo, como poder simular o comportamento do processo a decorrer e, se necessário, optimiza-lo.

Esta interface permite ainda que estas simulações e optimizações sejam efectuadas sem o processo real estar efectivamente a decorrer e que seja verificado se o processo pretendido é viável com as condições de operação disponíveis para esse efeito.

# 8 Trabalho futuro

No seguimento do trabalho efectuado é importante desenvolver uma interface mais versátil. A interface desenvolvida neste trabalho apenas permitia alterar os valores das concentrações, das constantes das isotérmicas e dos tempos de cada passo.

Industrialmente é também bastante importante ter a possibilidade efectuar alterações nos caudas de eluente e de alimentação a inserir no sistema, uma vez que se não estiver a ser utilizada uma proporção eluente/alimentação adequada pode levar a gastos desnecessários (no caso de se utilizar demasiado eluente) ou a que a separação não seja efectuada com sucesso.

Será também interessante conseguir ter uma visualização dinâmica tanto da monitorização em tempo real como das simulações em gProms de forma sincronizada de modo a se conseguir verificar quando existem alterações significativas no processo.

Para além dos perfis de concentração ao longo do tempo é importante ter perfis de concentração ao longo das colunas, que pode permitir verificar anomalias de funcionamento de uma determinada coluna.

Actualmente é necessário inicializar manualmente a aplicação gProms para se obter as simulações. Futuramente o objectivo seria que a própria interface tivesse a possibilidade de inicializar automaticamente e sempre que necessário a aplicação gProms. Isto toma grande importância pois a aplicação gProms é uma aplicação bastante complexa e, caso o utilizador conseguisse ter total controlo através da interface, não necessitaria de ter conhecimento das aplicações que a esta se encontram associadas.

A interface desenvolvida para comunicação por Shared Memory apenas pode ser utilizada para trocar informação dentro do mesmo computador. De modo a que possa

ser possível a um utilizador aceder a essa informação remotamente, um protocolo de comunicação por TCP/IP seria uma alternativa para a utilização e versatilização desta ferramenta de trabalho.

# 9 Bibliografia


[01] Tachibana, K., & Ohnishi, A. (2001). *Journal of Chromatography* , 906.

[02] Junior, I., Veredas, V., Santos, M., & Santana, C. (2006). Cromatografia em leito móvel simulado na produção de substâncias. *Chem. Nova* , Vol 29 nº5, 1027-1037.

[03] Weintraub, R. (19 de Janeiro de 2010). Continuous Chromatography: Simulated Moving Bed Systems. *SED Group Meeting Presentation* .

[04] Rodrigues, R. (2009). Compact SMB Chromatography for Binary Separation, PhD Thesis. Universidade Nova de Lisboa.

[05] Adam, P., Nicoud, R., Bailly, M., & Ludemann-Hombourger, O. (2000). *Patente N.º 6136198.* EUA.

[06] Ludemann-Hombourger, O., Pigorini, G., Nicoud, R., Ross, D., & Terfloth, G. (2002). Application of the "varicol" process to the separation of the isomers of the sb-553261 racemate. *Journal of Chromatography A 947* , 59-68.

[07] Toumi, A., Hanisch, F., & Engell, S. (2002). Optimal operation of continuous chromatographic processes: Mathematical optimization of the varicol process. *Industrial and Engineering Chemistry Research 41* , 4328–4337.

[08] Zhang, Z., Hidajat, H., Ray, A., & Morbidelli, M. (2002). Multiobjective optimization of smb and varicol process for chiral separation. *AIChE Journal 48* , 2800–2816.

[09] Pais, L., & Rodrigues, A. (2003). Design of simulated moving bed and varicol processes for preparative separations with a low number of columns. *Journal of Chromatography A 1006* , 33–44.

[10] Zhang, Z., Morbidelli, M., & Mazzotti, M. (2004). Experimental assessment of powerfeed. *AIChE Journal 50* , 625–632.

[11] Kearney, M., & Hieb, K. (1992). *Patente N.º 5102553.* EUA.

[12] Schramm, H., Kaspereit, M., Kienle, A., & Seidel-Morgenstern, A. (2003). Improved operation of simulated moving bed processes through cyclic modulation of feed flow and feed concentration. *Chemical Engineering Science 58* , 5217–5227.

[13] Schramm, H., Kaspereit, M., Kienle, A., & Seidel-Morgenstern, A. (2004). *Patente N.º 10235385.* Alemanha.

[14] Pais, L. S., Loureiro, J. M., & Rodrigues, A. E. (1998). Modeling strategies for enantiomers separation by smb chromatography. *AIChE Journal 44* , 561–569.

[15] Mazzotti, M., Storti, G., & Morbidelli, M. (1997). Optimal operation of simulated moving bed units for nonlinear chromatographic separations. *Journal of Chromatography A 769* , 3–24.

[16] Chin, C. Y., & LindaWang, N.-H. (2004). Simulated moving bed equipment designs. *Separation & Purification Reviews, 33* , 77-155.

[17] Lee, K. (2000). Two-section simulated moving-bed process. *Separation Science and Technology, 35* , 519–534.

[18] Jin, W., & Wankat, P. C. (2005). Two-zone smb process for binary separation. *Industrial & engineering chemistry research, 44* , 1565–1575.

[19] Rodrigues, R., Canhoto, T., Araújo, J., & Mota, J. (2008). Two-column simulated moving-bed process for binary separation. *Journal of Chromatography A 1180* , 42–52.

[20] Silva, R. (2013). Compact simulated countercurrent chromatography for downstream processing of (bio)pharmaceuticals. Universidade Nova de Lisboa.

[21] Klatt, K.-U., Hanische, F., Dünnebier, G., & Engell, S. Model-based optimization and control of chromatographic processes. Alemanha: Process Control Laboratory, Department of Chemical Engineering, University of Dortmund.

[22]. (s.d.). Obtido em Janeiro de 2014, de Microsoft: http://msdn.microsoft.com

[23]. (s.d.). *Michigan Tech University*. Obtido em Março de 2014, de http://www.cs.mtu.edu/vta/tcp-state-diagram.jpg

[24] Process Systems Enterprise Ltd. (Junho de 2004). gPROMS Introductory User Guide.

[25] Process Systems Enterprise Ltd. (Fevereiro de 2004). gPROMS Advanced User Guide.

[26] Process Systems Enterprise Ltd. (Outubro de 2004). gPROMS System Programmer Guide.

[27] Process Systems Enterprise Ltd. (Fevereiro de 2004). The gPROMS Server v2.3.

[28] Fourer, R., Gay, D. M., & Kernighan, B. W. (2002). AMPL: A Modeling Language for Mathematical Programming. *Duxbury Press* .

[29]. (s.d.). Obtido em Março de 2014, de AMPL: http://www.ampl.com/solvers.html

[30]. (s.d.). Obtido em Março de 2014, de IPOPT: https://projects.coin-or.org/Ipopt

[31]. (s.d.). Obtido em Março de 2014, de KNITRO: http://www.ziena.com/knitro.htm

[32] Byrd, R., Nocedal, J., & Waltz, R. (2006). *KNITRO: An Integrated Package for Nonlinear Optimization.*

[33] Mota, J. P., Esteves, I. A., M. Rostam-Abadi, M. (2004). C Dynamic modelling of an adsorption storage tank using a hybrid approach combining computational fluid dynamics and process simulation. *Computers and Chemical Engineering 28, 2421–2431*

[34] Araújo, J., Rodrigues, R., & Mota, J. P. (2006). Use of single-column models for efficient computation of the periodic state of a simulated moving-bed process. *Industrial and Engineering Chemistry Research 45* , 5314–5325.

[35] Araújo, J., & Mota, J. P. (2005). Single-column simulated-moving-bed process with recycle lag. *AIChE Journal* , 1641–1653.

[36] Araújo, J., Rodrigues, R., & Mota, J. P. (2006). Optimal design and operation of a certain class of asynchronous simulated moving bed processes. *Journal of Chromatography A 1132* , 76–89.